Titre: Title:	Recherche et analyse de codes convolutionnels doublement orthogonaux simplifiés au sens large
Auteur: Author:	Éric Roy
Date:	2006
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Roy, É. (2006). Recherche et analyse de codes convolutionnels doublement orthogonaux simplifiés au sens large [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/7830/

UNIVERSITÉ DE MONTRÉAL

RECHERCHE ET ANALYSE DE CODES CONVOLUTIONNELS DOUBLEMENT ORTHOGONAUX SIMPLIFIÉS AU SENS LARGE

ERIC ROY
DÉPARTEMENT DE GÉNIE ÉLECTRIQUE ET DE GÉNIE INFORMATIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES (M.Sc.A.)
(GÉNIE ÉLECTRIQUE)
AOÛT 2006

Your file Votre référence
ISBN: 978-0-494-25573-5
Our file Notre référence
ISBN: 978-0-494-25573-5

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé:

RECHERCHE ET ANALYSE DE CODES CONVOLUTIONNELS DOUBLEMENT ORTHOGONAUX SIMPLIFIÉS AU SENS LARGE

présenté par: ROY Eric

en vue de l'obtention du diplôme de: Maîtrise ès sciences appliquées

a été dûment accepté par le jury d'examen constitué de:

M. Jean-François Frigon, Ph.D., président

M. David Haccoun, Ph.D., membre et directeur de recherche

M. Christian Cardinal, Ph.D., membre et codirecteur de recherche

M. François Gagnon, Ph.D., membre

À mes parents, Claudette et Jean.

REMERCIEMENTS

Par ces quelques lignes je tiens à remercier les principaux acteurs qui m'ont permis, durant ces dernières années, de mener à bien cette recherche.

Premièrement, j'aimerais remercier mon directeur de recherche, le Dr. David Haccoun, ainsi que mon co-directeur de recherche, le Dr. Christian Cardinal, pour leurs précieux conseils ainsi que pour l'aide financière qu'ils m'ont accordés pendant cette recherche.

Je tiens aussi remercier tous mes collègues du laboratoire pour toutes ces discussions quotidiennes enjouées lors de ces deux dernières années. Je pense particulièrement à Zouheir, Oualid, Aniss, Nadia et Stéphane.

Finalement, je tiens à remercier ma famille ainsi que mes amis pour leur soutien.

RÉSUMÉ

La recherche proposée dans ce mémoire est un prolongement des travaux précédemment entamés sur les codes correcteurs d'erreur dits convolutionnels doublement orthogonaux (CSO^2C). L'utilisation de ce type de codes permet l'usage au récepteur d'une structure itérative de décodage sans entrelaceur. Cette propriété rend la combinaison des opérations de codage et de décodage très simples à réaliser en comparaison avec la technique de décodage itérative Turbo.

Toutefois, la recherche de codes CSO^2C se traduit en un problème d'optimisation très complexe consistant à trouver des codes répondant à toutes les conditions requises tout en minimisant leurs longueurs. Jusqu'à présent, les codes obtenus offrent des performances d'erreur très acceptables cependant, ils induisent un délai de décodage peu apprécié car la longueur des codes est très élévée.

Ainsi, l'objectif de ce travail de recherche fût de réduire la longueur des codes en éliminant certaines conditions imposées à ceux-ci. La prospection de ces nouveaux codes convolutionnels doublement orthogonaux dits simplifiés (S-CSO^2C-WS) et simplifiés perforés (S-PCSO^2C-WS) a permis d'importantes réductions de la longueur des codes conduisant à une diminution de la latence tout aussi prononcée au prix d'une légère dégradation des performances d'erreur.

Ce travail de recherche s'est avéré fructueux car il découvre et propose l'utilisation de nouveaux types de codes offrant de bons compromis fort avantageux entre la réduction de latence et la dégradation des performances d'erreur et ce, sans modifier la structure itérative utilisée lors du décodage.

ABSTRACT

This thesis presents a new class of error correcting codes called Simplified-Convolutional Self-Doubly Orthogonal codes (S-CSO^{2}C). These codes have the property that they can be easily decoded with an iterative threshold decoder without the need of interleavers.

Based on the algebraic properties of Convolutional Self-Doubly Orthogonal codes (CSO^{2}C), S-CSO^{2}C's are codes for which we have reduced some conditions on the double orthogonality of the CSO^{2}C codes. This action allows a important reduction of the span of the codes, which in return leads to a reduction of the total decoding delay. A Punctured version of S-CSO^{2}C's codes (S-PCSO^{2}C) is also investigated. We observed that the error performances was not decreasing dramatically, just a few tenth of a decibel at low signal to noise ratio, when we use simplified codes instead of CSO^{2}C codes. At high SNR, the error performances of simplified codes are the same as CSO^{2}C codes.

Finally, the use of S-CSO^{2}C's codes offers a good tradeoff between the error performances degradation and the time delay reduction at the decoder. For that, this study was fruitful and allows an easy implementation of this error control coding technique into real time communications systems.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

LISTE DES ANNEXES

LISTE DES SIGLES ET DES SYMBOLES

Sigles

BPSK	Binary Phase Shift Keying
BSC	Binary symmetric channel
CSOC	Convolutional Self-Orthogonal Code
CSO^{2}C	Convolutional Self-Doubly Orthogonal Code
CSO^{2}C-WS	Convolutional Self-Doubly Orthogonal Code-Wide Sense
CSO^{2}C-SS	Convolutional Self-Doubly Orthogonal Code-Strict Sense
EEPP	Equal Error Protection Punctured
LRV	Logarithme du rapport de vraisemblance
MAP	Maximum *A posteriori* Probability
S-CSO^{2}C-WS	Simplified Convolutional Self-Doubly Orthogonal Code-Wide Sense
S-PCSO^{2}C-WS	Simplified Punctured Convolutional Self-Doubly Orthogonal Code-Wide Sense
SCC	Self Convolutionnal Code
SNR	Signal to Noise Ratio
SOVA	Soft Output Viterbi Algorithm
UEPP	Unequal Error Protection Punctured

Symboles

E_b	Énergie du symbole binaire u_i
$\mathbf{v} = (v_1 \quad v_2 \quad \ldots \quad v_n)$	Vecteur de dimension n et ses composantes
m	Mémoire d'un code convolutionnel
R	Taux de codage
J	Nombre de connexions d'un code convolutionnel systématique de taux $R = \frac{1}{2}$
$\mathbf{P}$	Matrice de perforation
$L(\beta_i)$	Logarithme du rapport de vraisemblance de la variable aléatoire binaire β_i
$Q(x)$	$\frac{1}{\sqrt{2\pi}} \int_x^\infty \exp(-\frac{z^2}{2}) dz, \quad x \geq 0$
$\oplus$	Somme modulo 2
$\Diamond$	Opérateur add-min
$\lfloor x \rfloor$	Le plus grand entier inférieur à x
$\lceil x \rceil$	Le plus grand entier supérieur à x

CHAPITRE 1

INTRODUCTION

1.1 Motivations

La théorie de l'information établie vers la fin des années 1940, par C. E. Shannon a donné naissance entre autre à un théorème fondamental celui du codage de canal. Ce dernier expose les limites théoriques associées aux systèmes de communications utilisant le codage, comme moyen pour contrôler les erreurs de transmission provenant du canal. Malgré les résultats présentés par Shannon, rien n'indique comment nous devons nous y prendre pour atteindre les limites promises par la théorie.

Dès lors, une multitude de chercheurs et d'ingénieurs ont orienté leurs travaux dans le but d'élaborer des structures de codage et de décodage plus où moins complexes permettant de s'approcher de la terre promise par Shannon. Rapidement, le codage de canal a trouvé une niche au sein des communications numériques limitées en puissance telles que les communications spatiales.

Toutefois, ce n'est qu'en 1993 que Berrou, Glavieux et Thitimajshima proposèrent une structure de codage et de décodage bien particulière appelée codes Turbo qui, pour la première fois, permettait de s'approcher de la limite théorique de Shannon. Bien que cette technique offre d'excellentes performances d'erreur lorsque le canal est fortement bruité, quelques désavantages l'accompagne. D'une part, le délai associé au décodage peut être imposant et d'autre part, la complexité de mise en oeuvre est élevée. Dans le but de réduire cette complexité de mise en oeuvre ainsi que le délai imposé par l'opération de décodage, les professeurs Christian Cardinal, François Gagnon et David Haccoun ont proposé, en 1997, une technique

de codage et de décodage brevetée basée sur des propriétés algébriques des codes convolutionnels. Cette nouvelle classe de codes appelée codes convolutionnels doublement orthogonaux, permet l'utilisation d'un décodeur itératif à seuil en plus de permettre l'abolition des entrelaceurs que l'on retrouve dans la technique de codage Turbo. Ce nouveau système de codage permet de contourner les désavantages associés aux codes Turbo cependant, les performances d'erreur offertes par les codes doublement orthogonaux sont inférieures, lorsque le canal est très bruité, à celles obtenues avec les codes Turbo.

La problématique reliée aux codes doublement orthogonaux réside dans la recherche de bons codes. Les codes utilisés jusqu'à présent offrent de bonnes performances d'erreur, mais engendrent un délai de décodage assez important venant compromettre leur utilisation dans des systèmes de communications fonctionnant en temps réel. C'est ce problème qui vient motiver la recherche présentée dans ce mémoire.

Ainsi l'alternative proposée est l'utilisation d'une variante de codes doublement orthogonaux appelée codes doublement orthogonaux simplifiés. L'idée est de relaxer certaines conditions imposées aux codes doublement orthogonaux afin de les simplifier. De plus, dans le but d'obtenir facilement des codes de taux de codage élevés, la perforation des codes doublement orthogonaux simplifiés est aussi proposée.

1.2 Contributions

Les contributions apportées par ce travail de recherche sont les suivantes :

1. Élaboration de définitions associées aux codes convolutionnels doublement orthogonaux simplifiés au sens large et aux codes perforés simplifiés doublement orthogonaux au sens large.

2. Conception de programmes permettant la recherche de codes simplifiés doublement orthogonaux au sens large et de codes simplifiés perforés doublement orthogonaux au sens large.

3. Analyse et évaluation des performances d'erreur des codes simplifiés doublement orthogonaux au sens large et des codes simplifiés perforés doublement orthogonaux au sens large.

Toutes les simulations ont été effectuées à l'aide d'ordinateurs munis de processeurs pentium® IV d'Intel® cadencés à 3 Ghz et possédant 1 Gb de RAM sous l'environnement Windows XP®. Les logiciels de programmation Matlab® et Microsoft Visual Studio® ont servi à la conception des simulateurs utilisés.

1.3 Organisation du mémoire

Ce mémoire comporte cinq chapitres. À la suite du présent chapitre, le document se subdivise de la façon suivante.

- Le chapitre 2 présente les notions élémentaires associées aux codes convolutionnels orthogonaux. La structure de décodage à seuil des codes convolutionnels orthogonaux est présentée. Ensuite, nous exposons brièvement le principe associé à la technique de décodage turbo.

- Le chapitre 3 présente le décodeur itératif à seuil des codes convolutionnels doublement orthogonaux au sens large et définit les codes simplifiés doublement orthogonaux au sens large. Les algorithmes développés pour construire ces codes sont présentés et l'étude des performances des codes est entreprise. Enfin, les avantages ainsi que les inconvénients associés à ce nouveau type de codes sont dégagés.

- Le chapitre 4 résume le décodeur itératif à seuil pour les codes perforés et définit les codes perforés simplifiés doublement orthogonaux au sens large. Deux types de codes simplifiés et perforés sont présentés pour différents taux de codage. Les algorithmes utilisés pour la recherche de ces codes sont décrits ainsi que les performances d'erreur de ces codes.

- Finalement, le chapitre 5 résume l'ensemble des travaux effectués et propose certaines avenues pour lesquelles les codes simpliés pourraient être utilisés.

CHAPITRE 2

CODAGE DE CANAL

2.1 Introduction

Dans ce premier chapitre, nous abordons les différents concepts reliés aux codes convolutionnels systématiques orthogonaux (CSOC). Ainsi, nous analysons les structures de codage et de décodage relatives à ces codes ; lesquelles seront reprises aux chapitre 3 et 4. La dernière section, sur les codes Turbo, effectue le pont entre ce chapitre et les codes convolutionnels doublement orthogonaux (CSO^2C) présentés au chapitre 3.

2.1.1 Système de communications numériques

Dans cette section, nous présentons le système de communications numériques utilisé pour ce mémoire dont le schéma est exposé à la figure 2.1. Nous considérons une source qui émet une séquence d'information binaire $\mathbf{u} = (u_0, u_1, \ldots)$ où chaque bit u_i est d'énergie E_b. Cette séquence est ensuite transmise vers un codeur de canal de taux de codage R égal à k/n. Le taux de codage représente le rapport entre le nombre de bits d'information k nécessaires pour produire un mot de code de n bits. Le rôle du codeur est d'ajouter des symboles de redondance ($k < n$) selon certaines règles pour former la séquence binaire codée (ou le mot de code binaire) $\mathbf{v} = (v_0, v_1, \ldots)$. Cette séquence est ensuite envoyée vers un modulateur BPSK (modulation antipodale). Ce dernier transforme les symboles codés en symboles de canal $s_i(t)$, chacun d'énergie E_s, en utilisant la règle d'assignation suivante $s_i(t) = x_i f(t)$ où $x_i = (2v_i - 1)$ et où $f(t)$ représente une fonction définie sur le support $0 \leq t \leq T$ et d'énergie E_s.

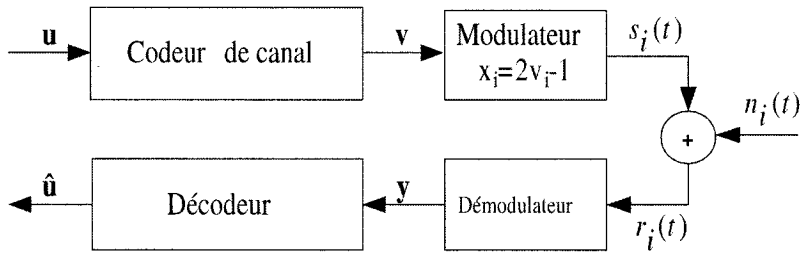

Figure 2.1: Schéma bloc du système de communication pour un canal BSC

Les symboles à la sortie du modulateur sont ensuite transmis vers le canal. Le canal considéré est de type binaire symétrique (BSC) sans mémoire à bruit blanc additif et gaussien. Par conséquent, $n_i(t)$ est un processus aléatoire gaussien de moyenne nulle et de densité de puissance spectrale bilatérale égale à $N_0/2$ (Watt/Hz). Le signal reçu $r_i(t) = s_i(t) + n_i(t)$ est démodulé de façon cohérente en utilisant un filtre adapté normalisé pour produire une sortie en valeur réelle y_i égale à :

$$y_i = \sqrt{E_s} x_i + n_i \tag{2.1}$$

où n_i représente une variable aléatoire gaussienne de moyenne nulle et de variance $N_0/2$. Le modèle équivalent en quantification ferme (le décodeur assigne une décision dure sur le symbole y_i) est donné par $\tilde{\mathbf{v}} = \mathbf{v} \oplus \mathbf{e}$ où le symbole binaire à la sortie du démodulateur à l'instant i vaut :

$$\tilde{v}_i = v_i \oplus e_i \tag{2.2}$$

où e_i est l'erreur binaire qui affecte le symbole codé v_i. La probabilité d'avoir une erreur $e_i = 1$, lorsqu'une modulation BPSK est utilisée pour transmettre l'information dans un canal BSC sans mémoire dont le bruit est additif et gaussien, est donnée par [17] :

$$\epsilon = \mathcal{Q}(\sqrt{\frac{2RE_b}{N_0}}) \tag{2.3}$$

et la probabilité de ne pas avoir d'erreur $e_i = 0$ est égale à $(1 - \epsilon)$. Les sorties non quantifiées $\mathbf{y} = (y_0, y_1, \ldots)$ ou dures $\tilde{\mathbf{v}} = (\tilde{v}_0, \tilde{v}_1, \ldots)$ sont ensuite envoyées vers le décodeur qui utilisera les symboles de redondance insérés par le codeur pour effectuer une estimation $\hat{u}_i$ sur les symboles d'information.

Pour ce type de canal, la capacité C établie par Shannon s'exprime à l'aide de la relation suivante [15]:

$$C = W \log_2 \left(1 + \frac{P_{moy}}{W N_0}\right) \quad \text{(bits/s)} \tag{2.4}$$

où W (Hz) et P_{moy} (Watt) représentent la largeur de bande disponible ainsi que la puissance moyenne d'émission respectivement. La signification de l'expression (2.4) est la suivante : si le débit d'information à la sortie de la source est inférieur à la capacité du canal C alors il est théoriquement possible, en utilisant un code correcteur d'erreur approprié, d'effectuer une transmission sans erreur. À l'opposé, si le débit de la source est supérieur à la capacité du canal alors quelque soit le code correcteur d'erreur utilisé, il ne sera pas possible de contrôler la fiabilité de la transmission. À partir de (2.4), nous pouvons déterminer le rapport signal sur bruit minimum $\left(\frac{E_b}{N_0}\right)_{min}$ à partir duquel il est possible de transmettre de l'information de manière fiable.

$$\left(\frac{E_b}{N_0}\right)_{min} \geq 0.69 \quad (-1.6\text{dB}) \tag{2.5}$$

Pour un canal très bruité où le rapport $\frac{E_b}{N_0}$ est inférieur à -1.6 dB il n'existe aucune technique de codage appropriée pour contrôler les erreurs. La valeur exprimée par (2.5) est généralement utilisée pour comparer les systèmes de codage entre eux.

2.2 Codes convolutionnels

Les codes convolutionnels se distinguent des codes en blocs, car les symboles codés à la sortie du codeur dépendent non seulement des symboles d'information à l'entrée, mais aussi des symboles d'information précédemment émis. Le codeur est essentiellement constitué de registres à décalage de longueurs finies reliés à des additionneurs modulo-2. Pour ce mémoire, nous nous intéresserons uniquement aux codes convolutionnels systématiques de taux de codage $R = \frac{1}{2}$.

Un code convolutionnel est dit systématique si la séquence à l'entrée du codeur est présente à l'une de ses sorties sans avoir été modifiée. La figure 2.2 présente un codeur convolutionnel systématique de taux de codage 1/2 où l'une des sorties $\mathbf{v^{(0)}}$ génère la séquence d'information à l'entrée $\mathbf{u}$. La seconde sortie $\mathbf{v^{(1)}}$ produit à chaque instant i un symbole de parité p_i associé au symbole d'information u_i. Ce symbole de parité est obtenu en effectuant l'addition modulo-2 entre certaines cellules du registre à décalage qui contiennent les symboles d'information précédemment émis. Par la suite, les deux séquences de sorties $\mathbf{v}^{(0)}$ et $\mathbf{v}^{(1)}$ sont

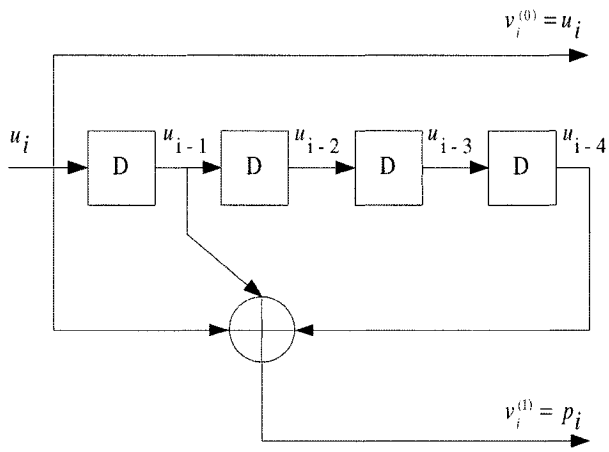

Figure 2.2: Codeur convolutionnel systématique de taux de codage 1/2, $m = 4$

multiplexées en une seule séquence de sorte que le mot de code transmis vers le canal est :

$$\begin{aligned} \mathbf{v} &= (v_0^{(0)}, v_0^{(1)}, v_1^{(0)}, v_1^{(1)}, \ldots) \\ &= (u_0, p_0, u_1, p_1, \ldots) \end{aligned}$$

L'ensemble des connexions reliant l'additionneur modulo-2 au registre à décalage est suffisant pour spécifier entièrement un code convolutionnel systématique. Cet ensemble $\mathcal{A}$ est définit par J nombres entiers positifs α_k représentant les positions des J connexions reliant le registre à l'additionneur modulo-2. Si la k-ième connexion relie la l-ième cellule du registre alors $\alpha_k = l$. La dernière connexion α_J de l'ensemble $\mathcal{A}$ représente la mémoire m[1] du codeur donc le nombre de symboles d'information stockés dans le registre . Nous pouvons, avec cet ensemble, représenter les symboles de parité aux instants $i = 0, 1, \ldots$ en fonction des éléments α_k, $k = 1, 2, \ldots, J$, de sorte que :

$$p_i = \sum_{k=1}^{J} \oplus u_{i-\alpha_k} \tag{2.6}$$

Exemple 2.1- L'ensemble des trois connexions associé au codeur de la figure 2.2 est donné par $\mathcal{A} = \{\alpha_1, \alpha_2, \alpha_3\} = \{0, 1, 4\}$, car la première connexion α_1 est reliée à l'entrée du codeur (position 0), la deuxième connexion α_2 est reliée à la première cellule et la dernière connexion α_3 est reliée à la quatrième cellule. La mémoire m de ce code est égale à quatre. Le symbole de parité à la sortie de ce codeur à l'instant i s'obtient par la somme $u_i \oplus u_{i-1} \oplus u_{i-4}$ qui vérifie l'équation (2.6).

---

[1]En fait, la mémoire d'un code est donnée par $\alpha_J - \alpha_1$. Cependant, nous considèrerons que les codes possédant la première connexion égale à 0, donc $m = \alpha_J$.

2.2.1 Codes convolutionnels systématiques orthogonaux

Les codes convolutionnels systématiques orthogonaux (CSOC) ont été introduit par Massey [13]. Leur structure permet un décodage algébrique simple que nous détaillons à la section 2.3. Pour qu'un code convolutionnel systématique de taux de codage 1/2 soit CSOC, l'ensemble $\mathcal{A}$ du code doit respecter la définition suivante.

> **Définition 1** - Un code convolutionnel systématique $\mathcal{A}$ de taux de codage 1/2 est orthogonal (CSOC) si les différences simples $s_{k,l} = (\alpha_k - \alpha_l)$, $k \neq l$, $\alpha_k, \alpha_l \in \mathcal{A}$ sont distinctes.

Pour un code $\mathcal{A}$ donné, nous noterons la collection de toutes les différences simples l'ensemble S. De cet ensemble nous distinguerons les différences simples positives $s_{k,l}$, $k > l$ qui forment le sous-ensemble S_+ des différences simples négatives $s_{k,l}$, $k < l$ qui forment le sous-ensemble S_-.

$$S = S_+ \cup S_-$$

En remarquant que pour chaque différence simple il existe la même différence simple, mais de signe opposé, $s_{k,l} = -s_{l,k}$, et par conséquent que les sous-ensembles S_+ et S_- contiennent les mêmes éléments à un signe près, il est donc suffisant de vérifier uniquement les différences simples positives pour tester la validité d'un code. La cardinalité du sous-ensemble S_+ sera notée N_s et est donnée par l'équation (2.7).

$$N_s = \mid S_+ \mid = \binom{J}{2} = \frac{J(J-1)}{2} \tag{2.7}$$

Pour illustrer la définition associée aux codes CSOC utilisons le codeur de la figure 2.2 dont l'ensemble des connexions est $\mathcal{A} = \{0, 1, 4\}$. Les différences simples positives engendrées par cet ensemble sont les suivantes : $(\alpha_3 - \alpha_1) = 4$, $(\alpha_3 - \alpha_2) = 3$, $(\alpha_2 - \alpha_1) = 1$. L'ensemble $S_+ = \{1, 3, 4\}$ contient trois différences simples distinctes par conséquent, le code $\mathcal{A}$ est orthogonal.

2.3 Décodeur à seuil

Nous présentons dans cette section un algorithme de décodage à logique majoritaire (où à seuil) pour les codes convolutionnels systématiques orthogonaux. L'artisan de cette technique de décodage, J. L. Massey, proposa cette structure [13] afin de concevoir un système correcteur d'erreur simple. Ce type de décodeur se différencie des algorithmes généralement utilisés pour décoder les codes convolutionnels, car il base ses décisions en fonction de considérations algébriques et non pas probabilistes comme l'algorithme de Viterbi, par exemple. Les deux décodeurs à seuil considérés dans cette section sont à sortie quantifiée et à sortie non-quantifiée.

2.3.1 Décodeur à seuil à sortie quantifiée

L'algorithme de décodage à seuil est fondé sur le calcul de syndrome. Pour chaque symbole d'information donné u_i le décodeur à seuil utilise uniquement l'ensemble des J syndromes relatifs aux bits de parité $\{p_{i+\alpha_k} : k = 1, 2, \ldots, J\}$ pour prendre une décision dure sur l'erreur $\hat{e}_i^u$ associée au symbole d'information u_i. En se référant au schéma du décodeur à seuil présenté à la figure 2.3, nous observons que pour générer les syndromes, le décodeur encode les symboles d'information provenant du canal $\tilde{u}_i$ pour former les symboles de parité p'_i. Ces symboles sont ensuite comparés avec les symboles de parité provenant du canal $\tilde{p}_i$ pour former les syndromes s_i.

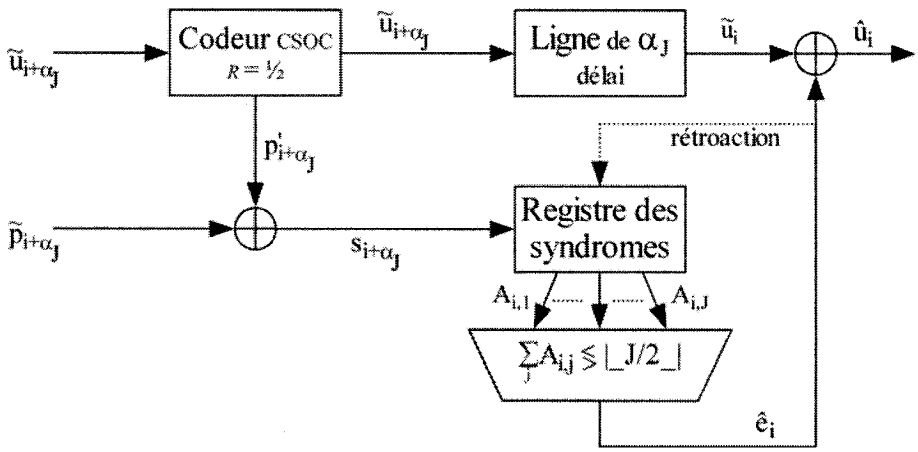

Figure 2.3: Schéma du décodeur itératif à seuil

Ainsi, le syndrome associé au bit de parité p_{i+k} peut s'écrire :

$$\begin{aligned} s_{i+k} &= \tilde{p}_{i+k} \oplus p'_{i+k} \\ &= e^p_{i+k} \oplus p_{i+k} \oplus \sum_{l=1}^{J} \oplus \tilde{u}_{i+k-\alpha_l} \\ &= e^p_{i+k} \oplus \sum_{l=1}^{J} \oplus u_{i+k-\alpha_l} \oplus \sum_{l=1}^{J} \oplus u_{i+k-\alpha_l} \oplus \sum_{l=1}^{J} \oplus e^u_{i+k-\alpha_l} \\ &= e^p_{i+k} \oplus \sum_{l=1}^{J} \oplus e^u_{i+k-\alpha_l} \end{aligned} \tag{2.8}$$

En mettant en facteur le terme e^u_i du terme de droite on obtient :

$$s_{i+k} = e^u_i \oplus e^p_{i+k} \oplus \sum_{l=1}^{k-1} \oplus e^u_{i+k-\alpha_l} \oplus \sum_{l=k+1}^{J} \oplus e^u_{i+k-\alpha_l} \tag{2.9}$$

Pour effectuer une estimation sur l'erreur $\hat{e}^u_i$ le décodeur doit calculer $(m+1)$ syndromes dont seuls les J syndromes $s_{i+\alpha_k}$ seront nécessaires pour prendre une décision sur le symbole décodé $\hat{u}_i$. Ces J syndromes forment l'ensemble des J équations de parité $\{A_{i,k}\,, k = 1, 2, \ldots, J\}$.

$$A_{i,k} = s_{i+\alpha_k} = e_i^u \oplus e_{i+\alpha_k}^p \oplus \sum_{l=1}^{k-1} \oplus e_{i+(\alpha_k-\alpha_l)}^u \oplus \sum_{l=k+1}^{J} \oplus e_{i+(\alpha_k-\alpha_l)}^u \tag{2.10}$$

Pour obtenir une estimation fiable sur l'erreur $\hat{e}_i^u$, certaines restrictions doivent êtres maintenues par l'ensemble des équations de parité. En fait, l'ensemble $\{A_{i,k}\}$ doit être orthogonal à l'erreur e_i^u. La propriété d'orthogonalité entre les J équations de parité est obtenue si toutes les équations de parité contiennent le symbole e_i^u et que tous les autres termes formant les équations sont distincts. En se référant à l'équation (2.10) nous pouvons observer qu'il est possible de former un tel ensemble orthogonal seulement si les différences $(\alpha_k - \alpha_l)$ sont distinctes. Les codes CSOC que nous avons présentés à la section 2.2.1 possèdent cette propriété d'où leur utilisation. Le décodeur effectue une estimation sur le symbole d'erreur $\hat{e}_i^u$ en utilisant une simple règle majoritaire (ou un seuil) de sorte qu'une erreur est détectée $\hat{e}_i^u = 1$ s'il y a plus de $\lfloor J/2 \rfloor$ équations de parités égales à 1 autrement $\hat{e}_i^u = 0$.

$$\hat{e}_i^u = \begin{cases} 0, & \sum_{k=1}^{J} A_{i,k} \leq \lfloor J/2 \rfloor \\ 1, & \sum_{k=1}^{J} A_{i,k} > \lfloor J/2 \rfloor \end{cases} \tag{2.11}$$

Une version de ce décodeur peut être obtenue en ajoutant une boucle de rétroaction [12], [13]. En remarquant que la dernière sommation de l'équation (2.10) est constituée de termes possédant des différences simples $(\alpha_k - \alpha_l)$ strictement négatives (donc la valeur de l'estimation $\hat{e}_{i+(\alpha_k-\alpha_l)}^u$ est connue à l'instant i) nous pouvons donc réintroduire ces termes dans les équations de parité pour minimiser l'erreur provenant des symboles $e_{i+\alpha_k-\alpha_l}^u$. Les équations $A_{i,k}$ s'écrivent alors :

$$A_{i,k} = e_i^u \oplus e_{i+\alpha_k}^p \oplus \sum_{l=1}^{k-1} \oplus e_{i+(\alpha_k-\alpha_l)}^u \oplus \sum_{l=k+1}^{J} \oplus (e_{i+(\alpha_k-\alpha_l)}^u \oplus \hat{e}_{i+(\alpha_k-\alpha_l)}^u) \tag{2.12}$$

Si nous considérons une rétroaction idéale, c'est-à-dire que l'estimation réinjectée

$\hat{e}^u_{i+(\alpha_k-\alpha_l)}$ s'avère toujours vraie, alors l'équation (2.12) devient :

$$A_{i,k} = e^u_i \oplus e^p_{i+\alpha_k} \oplus \sum_{l=1}^{k-1} \oplus e^u_{i+(\alpha_k-\alpha_l)} \tag{2.13}$$

Ce système d'équations est toujours orthogonal à l'erreur e^u_i et par conséquent, le décodeur choisira $\hat{e}^u_i = 1$ que si il y a plus de $\lfloor \frac{J}{2} \rfloor$ équations de parité égales à 1. Autrement, il choisira $\hat{e}^u_i = 0$. Les performances d'erreur associées à ce décodeur sont dictées par son seuil $\lfloor \frac{J}{2} \rfloor$ qui correspond au pouvoir de correction $t = \lfloor \frac{d_{min}-1}{2} \rfloor$. Le pouvoir de correction varie donc en fonction de la distance minimale d_{min} du code code CSOC utilisé:

$$d_{min} = J + 1 \tag{2.14}$$

Il s'en suit qu'augmenter le nombre de connexions α_k d'un code CSOC revient à en augmenter le pouvoir de correction. Ce type d'algorithme base ses décisions en fonction de la distance minimum du code ce qui en fait un algorithme sous optimum en comparaison avec l'algorithme de Viterbi, car $d_{min} \leq d_{free}$ [12]. On remarquera que pour deux codes CSOC de même dimension J, mais de mémoire $m = \alpha_J$ différente, on préférera, pour des raisons de conception, le code ayant la plus petite mémoire (ou longueur).

2.3.2 Décodeur à seuil à sortie non quantifiée

Le décodeur à seuil à sortie non quantifiée (APP Threshold Decoder) se différencie de son prédécesseur sous deux aspects. Premièrement, il produit une estimation directement sur le symbole décodé $\hat{u}_i$ et non plus sur l'erreur $\hat{e}^u_i$. Deuxièmement, il associe une valeur de fiabilité aux équations de parité en se basant sur le calcul des probabilités *a posteriori*.

Le nouveau système d'équations de parité utilisé par le décodeur $\{B_{i,k} : k = 0, 1, \ldots, J\}$ est composé de $(J + 1)$ équations obtenues à partir des équations de

parité orthogonales $A_{i,k}$. Les équations $B_{i,k}$ s'obtiennent en excluant le symboles $\tilde{u}_i$ de l'équation (2.12) et en créant l'équation supplémentaire $B_{i,0} = \tilde{u}_i$.

$$B_{i,k} = A_{i,k} \oplus \tilde{u}_i \tag{2.15}$$

$$= u_i \oplus e^p_{i+\alpha_k} \oplus \sum_{l=1}^{k-1} \oplus e^u_{i+\alpha_k-\alpha_l} \oplus \sum_{l=k+1}^{J} \oplus (e^u_{i+\alpha_k-\alpha_l} \oplus \hat{e}^u_{i+\alpha_k-\alpha_l}) \tag{2.16}$$

où $k = 0, 1, 2, \ldots, J$. Signalons que si l'ensemble $\{A_{i,k}\}$ est orthogonal au symbole d'erreur e^u_i alors l'ensemble $\{B_{i,k}\}$ est orthogonal au symbole u_i. Connaissant ce système d'équations orthogonales, le décodeur associera au symbole décodé $\hat{u}_i$ une mesure de fiabilité basée sur le calcul des probabilités *a posteriori*. Ceci revient à maximiser la probabilité conditionnelle $Pr\{u_i = V \mid \{B_{i,k}\}\}$ où V est une variable aléatoire binaire. Le décodeur choisira donc $\hat{u}_i = 1$ si :

$$Pr(u_i = 1 \mid \{B_{i,k}\}) \geq Pr(u_i = 0 \mid \{B_{i,k}\}) \tag{2.17}$$

autrement il choisira $\hat{u}_i = 0$. En prenant le logarithme de la relation (2.17) nous pouvons définir le logarithme du rapport de vraisemblance (LRV) de la variable aléatoire binaire u_i conditionnée sur l'ensemble des équations de parité orthogonales $\{B_{i,k}\}$ de sorte que le décodeur choisira $\hat{u}_i = 1$ si :

$$L(u_i \mid \{B_{i,k}\}) = \ln\left(\frac{Pr(u_i = 1 \mid \{B_{i,k}\})}{Pr(u_i = 0 \mid \{B_{i,k}\})}\right) \geq 0 \tag{2.18}$$

où la valeur absolue $|L(u_i \mid \{B_{i,k}\})|$ représente la fiabilité de la décision. La propriété d'orthogonalité entre les équations $B_{i,k}$ nous permet alors de considérer les symboles d'erreur comme étant indépendants (une erreur n'affecte qu'une équation à la fois). Cette considération nous permet de réécrire la relation (2.18) en utilisant le théorème de Bayes :

$$L(u_i \mid \{B_{i,k}\}) = \sum_{k=0}^{J} \ln\left(\frac{Pr(B_{i,k} \mid u_i = 1)}{Pr(B_{i,k} \mid u_i = 0)}\right) + \ln\left(\frac{Pr(u_i = 1)}{Pr(u_i = 0)}\right) \tag{2.19}$$

où $\ln\left(\frac{Pr(u_i=1)}{Pr(u_i=0)}\right)$ représente l'information *a priori* $L(u_i)$. Notons que si les probabilités *a priori* sont équiprobables alors le terme $L(u_i)$ devient nul. Nous pouvons simplifier l'expression (2.19) en observant que les équations $B_{i,k}$ prennent la valeur 1 que si le symbole $u_i = 0$ et que le nombre de symboles en erreur constituant l'équation de parité est impair ou bien si $u_i = 1$ et que le nombre de symboles en erreur est pair. Autrement, les équations $B_{i,k}$ prennent la valeur 0 si $u_i = 0$ et que le nombre de symboles en erreur dans les équations de parité est pair ou bien si $u_i = 1$ et que le nombre de symboles en erreur est impair. Ainsi, si nous définissons par $\gamma_{i,k}$ la probabilité d'avoir un nombre impair de symboles en erreur dans l'équation $B_{i,k}$ (en excluant le symbole u_i) alors $\rho_{i,k} = (1 - \gamma_{i,k})$ représente la probabilité d'avoir un nombre pair de symboles en erreur dans l'équation $B_{i,k}$. Donc,

$$Pr(B_{i,k} = 0 \mid u_i = 1) = Pr(B_{i,k} = 1 \mid u_i = 0) = \gamma_{i,k} \tag{2.20}$$

$$Pr(B_{i,k} = 0 \mid u_i = 0) = Pr(B_{i,k} = 1 \mid u_i = 1) = \rho_{i,k} \tag{2.21}$$

En regroupant les équations (2.20) et (2.21) dans (2.19) on obtient [13] :

$$L(u_i \mid \{B_{i,k}\}) = \sum_{k=1}^{J} (2B_{i,k} - 1)w_{i,k} + (2B_{i,0} - 1)w_{i,0} + L(u_i) \tag{2.22}$$

où $w_{i,k} = \ln\left(\frac{\rho_{i,k}}{\gamma_{i,k}}\right)$ représente la fiabilité (ou le poids) de l'équation de parité k et le terme $(2B_{i,k} - 1)$ représente le signe du symbole. En observant que $\gamma_{i,k} = Pr(B_{i,k} \oplus u_i = 1)$ et $\rho_{i,k} = Pr(B_{i,k} \oplus u_i = 0)$ nous pouvons réécrire l'expression des poids :

$$w_{i,k} = \ln\left(\frac{\rho_{i,k}}{\gamma_{i,k}}\right) = -\ln\left(\frac{Pr((B_{i,k} \oplus u_i) = 1)}{Pr((B_{i,k} \oplus u_i) = 0)}\right) = -L(B_{i,k} \oplus u_i) \tag{2.23}$$

Les poids peuvent donc êtres représentés par le LRV de la somme modulo-2 de deux variables aléatoires binaires indépendantes. En utilisant l'équation (2.16)

dans (2.23) on obtient :

$$w_{i,k} = -L\left(e^{p}_{i+\alpha_k} \oplus \sum_{l=1}^{k-1}\!\!\oplus e^{u}_{i+\alpha_k-\alpha_l} \oplus \sum_{l=k+1}^{J}\!\!\oplus (e^{u}_{i+\alpha_k-\alpha_l} \oplus \hat{e}^{u}_{i+\alpha_k-\alpha_l})\right) \tag{2.24}$$

Une simplification de ce LRV peut être apportée en utilisant l'opérateur *add-min* définit dans [11] et que nous représenterons par le symbole $\Diamond$. Cet opérateur effectue l'approximation du logarithme du rapport de vraisemblance d'une somme de n variables aléatoires binaires indépendantes β_i, $i = 1, 2, \ldots, n$ et peut s'exprimer par [11]:

$$L\left(\sum_{i=1}^{n}\!\!\oplus \beta_i\right) \approx \sum_{i=1}^{n}\!\!\Diamond L(\beta_i) = (-1)^{n+1}\prod_{i=1}^{n}\operatorname{sign}(L(\beta_i))\min_{1\le i\le n}\{\mid L(\beta_i)\mid\} \tag{2.25}$$

En combinant les relations (2.22), (2.24), (2.25) et en considérant l'information *a priori* nulle, il est montré dans [4] que l'approximation du LRV, $L(u_i \mid \{B_{i,j}\})$, notée λ_i peut s'écrire :

$$L(u_i \mid \{B_{i,j}\}) \approx \lambda_i = y^{u}_{i} + \sum_{k=1}^{J}\left(y^{p}_{i+\alpha_k} \Diamond \sum_{l=1}^{k-1}\!\!\Diamond y^{u}_{i+(\alpha_k-\alpha_l)} \Diamond \sum_{l=k+1}^{J}\!\!\Diamond \lambda_{i+(\alpha_k-\alpha_l)}\right) \tag{2.26}$$

$$= y^{u}_{i} + \sum_{k=1}^{J}\psi_{i,k} \tag{2.27}$$

$$= y^{u}_{i} + L_e(\hat{u}_i) \tag{2.28}$$

où le symbole $L_e(\hat{u}_i)$ représente l'information extrinsèque qui ne dépend pas de l'information reçue $\tilde{u}_i$. L'information extrinsèque ou supplémentaire représente la contribution de chacune des équations de parité sur la décision finale effectuée sur le symbole décodé $\hat{u}_i$. Le décodeur choisira donc :

$$\hat{u}_i = \begin{cases} 1 & \text{si } \lambda_i \ge 0 \qquad i = 0, 1, 2, \ldots \\ 0 & \text{autrement} \end{cases}$$

Exemple 2.2- Les équations de parité $\{\psi_{i,k}\}$ associées au code CSOC de la figure 2.2 peuvent s'écrire :

$$\begin{aligned}\psi_{i,1} &= y_i^p \Diamond \lambda_{i-1} \Diamond \lambda_{i-4} \\ \psi_{i,2} &= y_{i+1}^p \Diamond y_{i+1}^u \Diamond \lambda_{i-3} \\ \psi_{i,3} &= y_{i+4}^p \Diamond y_{i+4}^u \Diamond y_{i+3}^u\end{aligned}$$

de sorte que le schéma du décodeur à sortie pondérée relatif à cet exemple est donné par la figure 2.4.

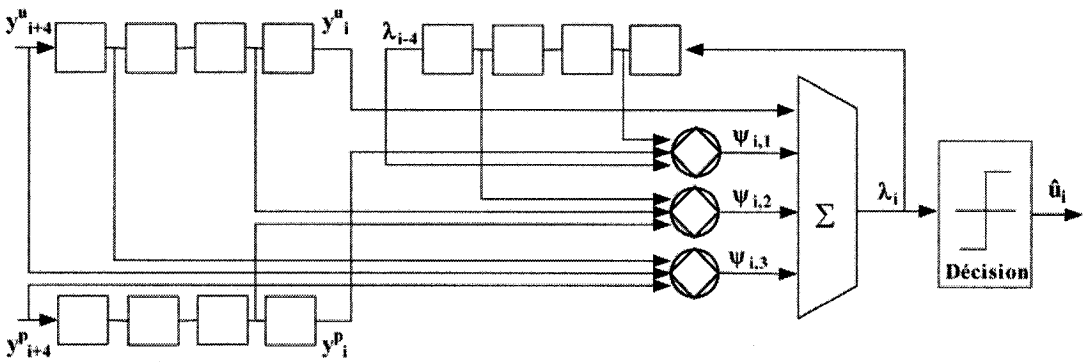

Figure 2.4: Décodeur à seuil à sortie pondérée associé au code $\mathcal{A} = \{0, 1, 4\}$

Remarquons que ce décodeur offre à sa sortie une valeur réelle (sortie pondérée) λ_i indiquant la fiabilité de la décision associée au symbole d'information décodé. C'est cette propriété qui permet l'utilisation du décodeur à seuil à sortie pondérée dans une structure itérative de décodage.

Le gain de codage asymptotique relatif au décodeur à seuil peut être exprimé par [12]:

$$G_c = 10\log_{10}(d_{\min} R) \quad \text{(dB)} \tag{2.29}$$

où $d_{\min}$ s'exprime selon (2.14).

2.4 Décodeur itératif *Turbo*

Le contrôle des erreurs à l'aide d'un décodeur itératif dit *Turbo* fut introduit en 1993 par Berrou, Glavieux et Thitimajshima [3]. Cette technique de décodage combinée à un codage concaténé en parallèle permet d'atteindre des performances d'erreur extrêmement près de la limite théorique établie par Shannon [15]. Les deux prochaines sections décrivent sommairement les opérations de codage et de décodage associées aux codes *Turbo*.

2.4.1 Codage *Turbo*

À la figure 2.5 nous présentons un codeur turbo de taux de codage 1/3 composé de deux codeurs convolutionnels en parallèle de faible mémoire. Les deux codeurs sont séparés par un module, l'entrelaceur (E), qui effectue la permutation des symboles d'information selon un certain ordre de sorte que les deux codeurs convolutionnels ne codent pas la même séquence d'information, $\mathbf{u} \neq \mathbf{u}'$. Cette dernière opération joue un rôle clé, car c'est elle qui rend les performances d'erreur aussi attrayantes. Pour obtenir de hautes performances d'erreur, la taille de l'entrelaceur doit être élevée, de l'ordre de plusieurs milliers de bits. Il s'en suit que plus la taille de l'entrelaceur est élevée, plus le délai de codage est important rendant l'utilisation de ce codeur difficile pour des communications en temps réel.

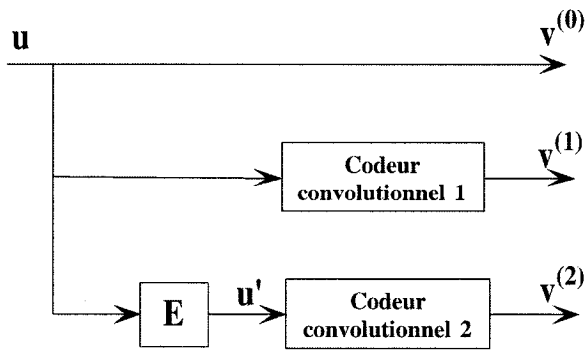

Figure 2.5: Schéma bloc du codeur *Turbo*

Par exemple, supposons qu'une source délivre des bits d'information à un taux de 8 Kbps et que le codeur utilise un entrelaceur de 64 Kbits. Le délai de codage induit par cet entrelaceur est de 8 secondes ! Ce délai serait intolérable lors d'une conversation téléphonique sans fil.

2.4.2 Décodage itératif *Turbo*

Le décodage itératif turbo repose sur l'échange d'information entre deux décodeurs en série. Cet échange permet l'amélioration de l'estimation sur les symboles provenant de la source. Le décodeur turbo présenté à la figure 2.6 est composé de deux décodeurs, d'un entrelaceur identique à celui utilisé par le codeur et d'un délaceur (le délaceur effectue l'opération inverse de l'entrelaceur).

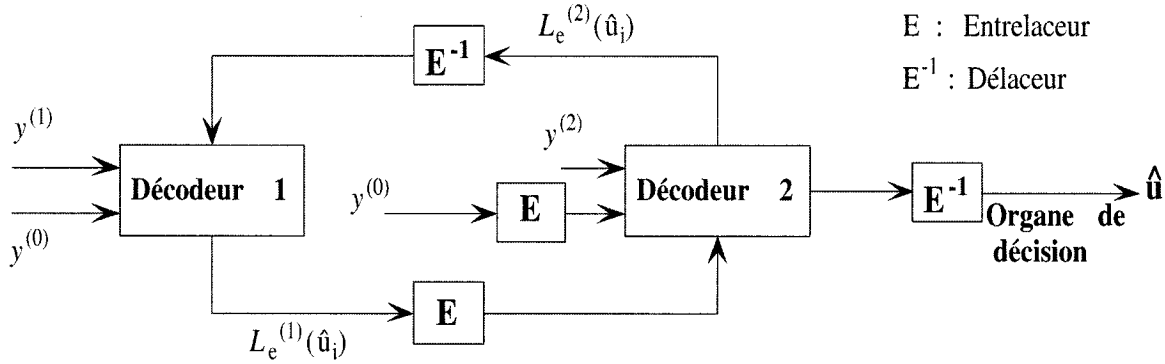

Figure 2.6: Schéma bloc du décodeur itératif *Turbo*

Généralement, les algorithmes utilisés par les deux décodeurs sont de type SOVA (Soft Output Viterbi Algorithm)[12] ou BCJR [2]. Ces algorithmes produisent une estimation (des valeurs de fiabilité) sur chaque symbole d'information. Par exemple, l'algorithme BCJR est une technique de décodage symbole par symbole calculant la valeur MAP (Maximum *a posteriori*) sur chaque symbole d'information. En se référant à la figure 2.6, l'échange de l'information entre les deux décodeurs s'effectue de la façon suivante. À l'état initial, le premier

décodeur calcule la valeur extrinsèque $L_e^{(1)}(\hat{u}_i)$ à partir des sorties pondérées du canal associée aux sorties $v_i^{(0)}$ et $v_i^{(1)}$ du codeur. Cette valeur est transmise vers le second décodeur en passant par l'entrelaceur. Le deuxième décodeur utilisera $L_e^{(1)}(\hat{u}_i)$ comme information *a priori* ainsi que les sorties pondérées du canal associées aux sorties $v_i^{(0)}$ et $v_i^{(2)}$ du codeur pour produire à sa sortie l'information extrinsèque $L_e^{(2)}(\hat{u}_i)$ qui sera retournée vers le premier décodeur en passant par le délaceur. Le premier décodeur utilisera l'information extrinsèque $L_e^{(2)}(\hat{u}_i)$ comme information a priori en plus des symboles provenant du canal pour produire $L_e^{(1)}(\hat{u}_i)$. Ce cycle représente une itération et est répété jusqu'à la convergence de l'algorithme ou jusqu'à ce que le nombre d'itérations maximum soit atteint. Le rôle de l'entrelaceur dans ce processus est de permettre l'indépendance entre les observables d'une itération à l'autre pour améliorer l'estimation sur les symboles d'information.

Les performances exceptionnelles qu'offrent les techniques de codage et de décodage turbo reposent essentiellement sur la taille de l'entrelaceur et sur le nombre d'itérations effectuées lors du décodage. Initialement, la structure proposée [3] utilisait deux codeurs de mémoire $m = 4$, un entrelaceur pseudo-aléatoire de taille 65536 bits et deux décodeurs utilisant l'algorithme MAP. En effectuant 18 itérations, cette configuration permet d'atteindre une probabilité d'erreur par bit de 10^{-5} à 0.7 dB de la capacité du canal. Ces résultats font en sorte que cette technique offre d'excellentes performances d'erreur lorsque le canal est très bruité. Toutefois, les performances d'erreur atteignent un certain "planché d'erreur" (error floor), l'algorithme a atteint pour des valeurs de $\frac{E_b}{N_0}$ croissantes sa valeur de convergence pratique et ne permet plus d'améliorer les probabilités d'erreur même avec un plus grand nombre d'itération. Ceci dit, pour des systèmes de communications numériques nécessitant une faible probabilité d'erreur (sous le planché d'erreur) cette technique de codage peut devenir un problème.

Notons que le prix à payer pour obtenir de telles performances est important. D'une part, le délai de décodage est imposant, car il est proportionnel au nombre d'itérations ainsi qu'à la taille de l'entrelaceur. D'autre part, la mise en oeuvre de cette technique est très complexe et varie selon l'algorithme de décodage utilisé. Dans le but de réduire la complexité de mise en oeuvre ainsi que le délai de décodage, les auteurs de [5] et [10] ont proposé une technique inspirée du concept apporté par le décodage itératif turbo et du concept d'orthogonalité des codes orthogonaux. Cette nouvelle structure est présentée dans le prochain chapitre.

CHAPITRE 3

DÉCODEUR À SEUIL ITÉRATIF ET CODES CONVOLUTIONNELS DOUBLEMENT ORTHOGONAUX SIMPLIFIÉS AU SENS LARGE (S-CSO^{2}C-WS)

3.1 Introduction

Dans ce chapitre, nous exposons les principes du décodage à seuil itératif et nous définissons une nouvelle classe de codes pouvant êtres utilisés avec ce type de décodeur, soient les codes convolutionnels doublement orthogonaux simplifiés au sens large (S-CS0^{2}C-WS).

Le décodeur à seuil itératif proposé dans [4] est une technique de décodage itérative, symbole par symbole, permettant l'abolition des entrelaceurs que l'on retrouve dans la technique de codage *Turbo*. Toutefois, pour compenser le rôle prépondérant joué par ces entrelaceurs, qui assurent l'indépendance entre les observables d'une itération à l'autre, les codes convolutionnels systématiques utilisés dans cette nouvelle structure doivent répondre à certaines conditions dites de double orthogonalité.

La recherche de ces codes se traduit alors en un problème d'optimisation très complexe consistant à trouver des codes convolutionnels systématiques répondant à toutes les conditions requises, tout en minimisant leur longueur. À ce jour, les codes trouvés [1], [7] et [8] possèdent tous une longueur de contrainte élevée induisant alors un délai de décodage important compromettant en quelque sorte, le principal avantage de cette méthode de décodage. Pour faciliter la recherche des codes et ainsi réduire leur longueur (span), nous avons réduit certaines contraintes imposées à ceux-ci et nous avons étudié l'impact de cette réduction sur les performances d'erreur.

3.2 Décodeur à seuil itératif

Le décodeur à seuil itératif utilisé est constitué d'une succession de M décodeurs à seuil à sortie non quantifiée, tout comme celui décrit à la section 2.3.2. La figure 3.1 présente le shéma bloc du décodeur considéré. Son fonctionnement est le suivant : à chaque itération μ, $1 \leq \mu \leq M$, le décodeur calcule la sortie pondérée $\lambda_i^{(\mu)}$ associée au symbole u_i, en se basant sur les symboles provenant du canal, y_i^u et y_i^p, et de la sortie pondérée provenant de l'itération précédente $\lambda_i^{(\mu-1)}$. À la dernière itération $\mu = M$, le décodeur compare la valeur de fiabilité $\lambda_i^{(M)}$ à un seuil afin d'effectuer une décision sur le symbole d'information transmis. La règle de décision est donnée par :

$$\hat{u}_i = \begin{cases} 1, & \text{si} \quad \lambda_i^{(M)} \geq 0 \\ 0, & \text{si} \quad \lambda_i^{(M)} < 0 \end{cases} \tag{3.1}$$

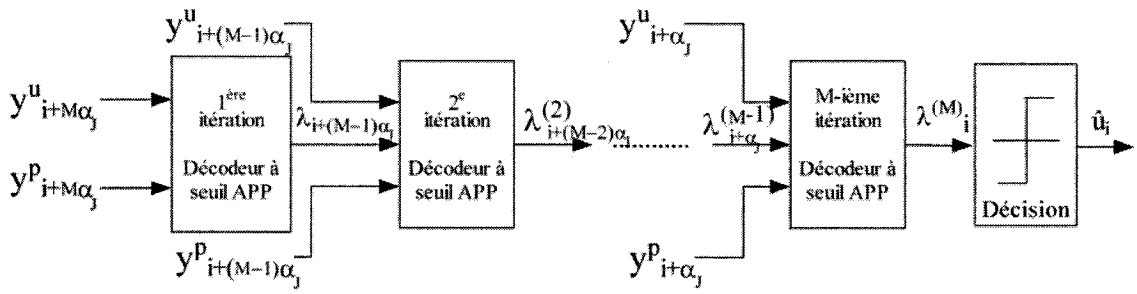

Figure 3.1: Schéma bloc du décodeur itératif à seuil

où la sortie non quantifiée à la M-ième itération s'écrit [4] :

$$\begin{aligned} \lambda_i^{(M)} &= y_i^u + L_e^{(M)}(\hat{u}_i) \\ &= y_i^u + \sum_{k=1}^{J} \psi_{i,k}^{(M)} \\ &= y_i^u + \sum_{k=1}^{J} (y_{i+\alpha_k}^p \Diamond \sum_{l=1}^{k-1}\!\!\Diamond \lambda_{i+\alpha_k-\alpha_l}^{(M-1)} \Diamond \sum_{l=k+1}^{J}\!\!\Diamond \lambda_{i+\alpha_k-\alpha_l}^{(M)}) \end{aligned} \tag{3.2}$$

L'équation (3.2) nous permet de constater que pour maintenir l'indépendance entre les observables tout au long du processus de décodage, les codes utilisés ne doivent plus répondre seulement à la simple condition d'orthogonalité énoncée à la section 2.2.1. Ils doivent plutôt êtres M-orthogonaux, c'est-à-dire que l'orthogonalité doit être maintenue entre les équations, $\psi_{i,k}^{(\mu)}$, et ce pour les M itérations. Cependant, il est montré dans [4] que la condition de double orthogonalité est suffisante pour atteindre les performances d'erreur asymptotiques obtenues à l'aide des codes M-orthogonaux.

Le délai de décodage total τ associé au décodeur présenté à la figure 3.1 est proportionnel à la longueur des codes convolutionnels utilisés α_J et au nombre d'itérations effectuées. Nous pouvons écrire :

$$\tau = M\alpha_J \quad \text{(bits)} \tag{3.3}$$

où M et α_J représentent le nombre d'itérations et la longueur du code utilisé respectivement. Il s'en suit qu'une minimisation de la longueur des codes entraîne une diminution du délai de décodage.

3.3 Codes convolutionnels doublement orthogonaux au sens-large (CSO^{2}C-WS)

On peut formaliser les conditions de la double orthogonalité des codes en observant la somme des termes composant l'information extrinsèque à la seconde itération. Cette somme est donnée par $\sum_{k=1}^{J} \psi_{i,k}^{(2)}$, de sorte qu'en combinant les relations (2.26) et (3.2) on obtient [4]:

$$\begin{aligned}\psi_{i,k}^{(2)} = y_{i+\alpha_k}^{p} \Diamond \sum_{l=k+1}^{J} \Diamond \lambda_{i+\alpha_k-\alpha_l}^{(2)} \Diamond \sum_{l=1}^{k-1} \Diamond \Big(y_{i+\alpha_k-\alpha_l}^{u} + \sum_{n=1,n\neq l}^{J} \Big(y_{i+\alpha_k-\alpha_l+\alpha_n}^{p} \Diamond \\ \sum_{m=1,m\neq k}^{n-1} \Diamond y_{i+(\alpha_k-\alpha_l)-(\alpha_m-\alpha_n)}^{u} \Diamond \sum_{m=n+1}^{J} \Diamond \lambda_{i+(\alpha_k-\alpha_l)-(\alpha_m-\alpha_n)}^{(1)} \Big) \Big)\end{aligned} \tag{3.4}$$

À partir de (3.4) nous pouvons définir les conditions nécessaires sur l'ensemble des connexions $\{\alpha_k\}$ d'un code pour que les observables soient indépendants sur les deux premières itérations. On définit alors les codes convolutionnels doublement orthogonaux au sens large comme suit [4]:

> **Définition 1** : Un code convolutionnel systématique de taux de codage $R = \frac{1}{2}$ est doublement orthogonal au sens large (CSO^{2}C-WS) si et seulement si l'ensemble des connexions $\mathcal{A} = \{\alpha_k \ : \ k = 1, 2, \ldots, J\}$, répond aux trois conditions suivantes :
>
> **1-** L'ensemble des différences simples $S = \{s_{k,l} = (\alpha_k - \alpha_l) : k \neq l\}$ est composé d'éléments *distincts* ;
>
> **2-** L'ensemble des différences des différences $D = \{d_{m,n}^{k,l} = (\alpha_k - \alpha_l) - (\alpha_m - \alpha_n) : k \neq l, m \neq n, k \neq m, l \neq n\}$ est composé d'éléments *distincts* à l'exception des différences des différences inévitables provenant de la permutation des indices (l, m) et (k, n);
>
> **3-** Les ensembles des différences simples et des différences des différences sont *disjoints*, $S \cap D = \emptyset$.

Nous remarquons avec cette définition que tous les termes composant les équations $\psi_{i,k}^{(2)}$ $k = 1, 2, \ldots, J$ sont différents les uns des autres à l'exception des différences doubles[1] inévitables provenant de la permutation possible des indices (l, m) et (k, n) dans l'équation (3.4). Ces différences doubles non distinctes rendent les équations $\psi_{i,k}^{(2)}$ corrélées et viennent ainsi compromettre l'indépendance des symboles dans les équations de parité [2], de sorte que l'estimation des symboles n'est pas aussi fiable que voulue. Comme l'ensemble des différences doubles D comporte des différences doubles inévitables, on peut considérer selon [8] l'ensemble des différences doubles sans les inévitables D^* (on exclut les différences doubles inévitables de l'ensemble D). Par conséquent, pour un codes CSO^{2}C-WS les éléments constituant l'ensemble D^* sont tous *distincts* et l'ensemble D^* est donnée par :

$$D^* = \{d_{m,n}^{k,l} = (\alpha_k - \alpha_l) - (\alpha_m - \alpha_n) : k \neq l,\ m \neq n,\ k \neq m,\ l \neq n,\ k \geq n,\ l \geq m\}$$

Tous les éléments constituant l'ensemble D^* sont positifs ou négatifs ce qui nous permet de réécrire cet ensemble à l'aide des sous-ensembles D_+^* et D_-^* qui contiennent les différences doubles positives et négatives respectivement.

$$D^* = D_+^* \cup D_-^*$$

La cardinalité de D_+^* notée N_d est donné par [8]:

$$N_d = \mid D_+^* \mid = \mid D_-^* \mid = \frac{1}{8}(J^4 - 2J^3 + 3J^2 - 2J) \tag{3.5}$$

[1] Pour faciliter la lecture du mémoire, les différences des différences seront appelées différences doubles.

[2] À titre indicatif, nous mentionnons qu'il est possible d'éviter les différences doubles inévitables en utilisant une autre classe de codes convolutionnels doublement orthogonaux définis au sens strict. Toutefois, cette classe de codes ne sera pas détaillée dans ce mémoire et le lecteur est référé à [4].

Les auteurs de [9] ont identifié, à l'aide de graphes, le rôle que joue chacune des conditions sur la prise de décision effectuée par le décodeur itératif. De cette analyse, nous pouvons tirer les conclusions suivantes. La condition 1 (les différences simples doivent être distinctes) est la plus importante à respecter, car c'est elle qui a le plus d'influence sur la prise de décision lors du décodage. Ensuite, les conditions 3 et 2 sont celles qui doivent être respectées par ordre d'importance. À partir de ces résultats, si nous désirons simplifier les conditions imposées aux codes CSO^2C-WS, mieux vaut alors réduire la condition ayant le moins d'impact lors du décodage soit la deuxième. Cette conclusion, nous permet de définir, à la section suivante, les codes simplifiés doublement orthogonaux.

3.4 Définition des codes convolutionnels doublement orthogonaux simplifiés au sens-large (S-CSO^{2}C-WS)

Les codes convolutionnels doublement orthogonaux simplifiés au sens large (S-CSO^{2}C-WS) représentent une classe de codes pour lesquels nous avons relaxé la condition exigeant que les différences doubles doivent êtres distinctes. Autrement dit, pour les codes simplifiés, les différences doubles inévitables ne constituent pas les seules différences doubles égales entre elles. Contrairement au codes CSO^{2}C-WS, l'ensemble des différences doubles positives sans les inévitables D_+^* contient N_d^e éléments identiques. Nous définissons alors les codes doublement orthogonaux simplifiés au sens large (S-CSO^{2}C-WS) selon la définition suivante.

Définition 2 : Un code convolutionnel systématique de taux de codage $R = \frac{1}{2}$ est doublement orthogonal et simplifié (S-CSO^{2}C-WS) si et seulement si l'ensemble des connexions $\mathcal{A} = \{\alpha_k \ : \ k = 1, 2, \ldots, J \}$ répond aux conditions suivantes :

1- L'ensemble des différences simples $S = \{s_{k,l} = (\alpha_k - \alpha_l) : k \neq l\}$ est composé d'éléments *distincts* ;

2- L'ensemble des différences des différences $D = \{d_{m,n}^{k,l} = (\alpha_k - \alpha_l) - (\alpha_m - \alpha_n) \ : \ k \neq l, m \neq n, k \neq m, l \neq n\}$ est composé de $2N_d^e$ éléments *égaux* en excluant les différences des différences inévitables provenant de la permutation des indices, $N_d^e < N_d$;

3- Les ensembles des différences simples et des différences des différences sont *disjoints*, $S \cap D = \emptyset$.

3.4.1 Facteur de simplification associé aux codes S-CSO^{2}C-WS

Nous caractérisons les codes simplifiés par un facteur, δ, qui indique le degrés de non conformité d'un code par rapport à la définition 1 associée aux codes CSO^{2}C-WS. Ce facteur représente le rapport entre les différences doubles égales positives N_d^e (le nombre d'éléments positifs identiques dans D_+^*) et le nombre total de différences doubles positives (le nombre d'éléments positifs $N_d = \mid D_+^* \mid$ dans l'ensemble D^*). On rappelle que l'ensemble D^* représente l'ensemble des différences doubles D excluant les différences doubles inévitables. Le rapport de simplification s'écrit donc :

$$\delta = \frac{N_d^e}{N_d} \ , 0 \leq \delta < 1 \tag{3.6}$$

où N_d est donné par l'équation (3.5). Si le rapport δ est égal à 0, alors le nombre de différences doubles égales est nul et le code est doublement orthogonal au sens large. Il est possible de formuler une borne supérieure[3] δ^* sur le facteur de simplification d'un code selon l'équation (3.7) :

$$\delta^* = 1 - \frac{N_s}{N_d} \tag{3.7}$$

où N_s représente le nombre de différences simples positives. Nous devons noter que les différences doubles égales ne se répartissent pas de façon uniforme entre les équations de parité rendant certaines d'entre elles plus corrélées que d'autres. Dans ce mémoire, on ne tient pas compte de cette distribution des différences doubles égales entre les équations de parité.[4]

[3]La démonstration de cette borne est énoncée à la section I.2 de l'annexe I.

[4]Voir l'annexe III pour un exemple de la répartition des différences doubles égales entre les équations de parité.

Exemple 3.1- Soit le code CSOC, $\mathcal{A} = \{0, 1, 4\}$, utilisé lors des exemples précédents où l'ensemble des différences simples positives est donné par $S_+ = \{1, 3, 4\}$. L'ensemble complet des différences doubles sans les différences doubles inévitables, D^*, au sens des définitions 1 et 2 est donné par :

$$\begin{array}{lll} d_{2,1}^{1,2} = -2 & d_{1,2}^{2,3} = -2 & d_{1,2}^{3,1} = 5 \\ d_{2,1}^{1,3} = -5 & d_{3,2}^{2,3} = -6 & d_{1,3}^{3,1} = 8 \\ d_{3,1}^{1,3} = -8 & d_{3,1}^{2,3} = -7 & d_{2,3}^{3,2} = 6 \\ d_{1,2}^{2,1} = 2 & d_{1,3}^{3,2} = 7 & d_{2,1}^{3,2} = 2 \end{array}$$

$$\begin{aligned} D^* &= \{-2, -2, -5, -6, -7, -8\} \cup \{2, 2, 5, 6, 7, 8\} \\ &= D_-^* \cup D_+^* \end{aligned}$$

donc $D_+^* = \{2, 2, 5, 6, 7, 8\}$ et $N_d = \mid D_+^* \mid = \frac{J^4 - 2J^3 + 3J^2 - 2J}{8} \mid_{J=3} = 6$. On remarque la présence de différences doubles identiques dans l'ensemble D_+^* et par conséquent le code n'est pas de type CSO^{2}C-WS, car la condition 2 de la définition 1 n'est pas respectée. On vérifie que les ensembles D_+^* et S_+ sont disjoints donc l'ensemble $\mathcal{A}$ répond aux conditions des codes S-CSO^{2}C-WS. Le nombre d'éléments identiques N_d^e dans l'ensemble D_+^* est de un, donc le rapport de simplification est donné par $\delta = \frac{1}{N_d} = \frac{1}{6}$.

3.4.2 Choix des coefficients de pondération

L'introduction d'éléments corrélés dans les équations de parité $\psi_{i,k}^{(\mu)}$ dû aux permutations inévitables des indices et à la simplification des conditions biaise l'estimation sur le symbole à décoder et entraîne une dégradation non négligeable des performances d'erreur. Pour combattre ces effets nuisibles, une modification peut être apportée au décodeur itératif à seuil, de sorte que les équations de parité sont pondérées à l'aide d'un coefficient. Nous pouvons réécrire (3.2) de sorte que les sorties pondérés deviennent [4] :

$$\lambda_i^{(\mu)} = a_0^{(\mu)} y_i^u + \sum_{k=1}^{J} a_k^{(\mu)} \psi_{i,k}^{(\mu)} \tag{3.8}$$

où $a_k^{(\mu)}$, $k = 0, 1, \ldots, J$ représente le coefficient de pondération à l'itération μ associé à la k-ième équation de parité. Le but de cette opération est de "décorréler" les équations en quelque sorte, de façon à minimiser la probabilité d'erreur. Le problème consiste donc à trouver la distribution des poids qui minimisera la probabilité d'erreur. Il est montré dans [4] et [6] que de bons résultats sont obtenus en considérant un coefficient de pondération constant pour toutes les itérations, $a_k^{(\mu)} = a^*$, ce qui simplifie le calcul des valeurs $\lambda_i^{(\mu)}$. Par conséquent (3.8) devient :

$$\lambda_i^{(\mu)} = a^* \left(y_i^u + \sum_{k=1}^{J} \psi_{i,k}^{(\mu)} \right) \tag{3.9}$$

La détermination des coefficients de pondération s'effectue à l'aide de simulations pour lesquelles nous fixons le rapport E_b/N_0 et où nous faisons varier le coefficient de pondération pour toutes les itérations. Le tableau 3.1 présente les meilleurs coefficients de pondération[5] pour les codes simplifiés et CSO^{2}C-WS. On observe que quelque soi le type de code utilisé, pour une dimension des codes fixée, le coefficient de pondération demeure le même.

[5]Nous pouvons retrouver ces résultats à l'annexe II.

Tableau 3.1: Meilleurs coefficients a^* pour les codes les plus simplifiés S-CSO^{2}C-WS et CSO^{2}C-WS, $6 \leq J \leq 11$, $E_b/N_0 = 4$ dB

		J					
		6	7	8	9	10†	11†
a^*	S-CSO^{2}C	0.30	0.30	0.25	0.20	0.20	0.20
	CSO^{2}C	0.30	0.30	0.25	0.20	0.20	0.20

†Pour $10 \leq J \leq 11$, $E_b/N_0 = 3$ dB

3.5 Recherche des codes simplifiés S-CSO^{2}C-WS

3.5.1 Algorithmes de recherche

Dans cette section, nous présentons les deux méthodes de recherche utilisées pour construire les codes simplifiés doublement orthogonaux au sens large.

- **Recherche exhaustive**

Le premier algorithme de recherche présenté est l'algorithme de recherche exhaustif. Il est simple à concevoir et nous garantit un balayage complet de l'arbre de recherche. Il nous offre donc une solution optimale quant à la minimisation de la longueur des codes. Le schéma bloc à la figure 3.2 présente cet algorithme de recherche.

Le fonctionnement de l'algorithme est le suivant : une nouvelle connexion est créée si les connexions précédentes répondent à la définition des codes S-CSO^{2}C-WS. Cette connexion prend la valeur de la connexion précédente à laquelle nous ajoutons une unité. Nous comparons la valeur de cette connexion à une métrique, α^{max}, qui représente le code le plus court. À l'état initial, la métrique est initialisée à l'infini. Si l'exploration d'une branche se termine sans satisfaire les critères, alors un mécanisme de retour en arrière permet l'emprunt d'une nouvelle branche de recherche. La recherche se termine lorsque $\alpha_1 = 1$, car tous les codes S-CSO^{2}C-WS débutent par la valeur 0.

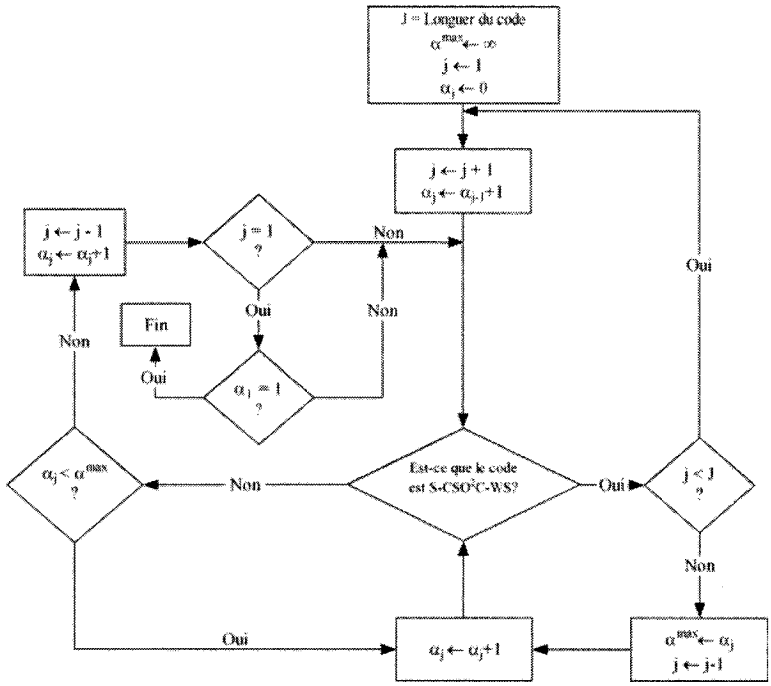

Figure 3.2: Diagramme représentant l'algorithme de recherche exhaustif

---

Exemple 3.2- Cet exemple montre le parcours effectué dans l'arbre de recherche par l'algorithme exhaustif pour rechercher le plus petit code S-CSO^{2}C-WS de dimension 3, $\mathcal{A} = \{\alpha_1, \alpha_2, \alpha_3\}$.

À l'état initial, la première connexion α_1 et la métrique de recherche α^{max} sont initialisées à zéro et l'infini respectivement. La recherche débute en créant une nouvelle connexion α_2 égale à $(\alpha_1 + 1)$. Ces deux connexions forment alors l'ensemble $\mathcal{A} = \{0, 1\}$ qui répond aux conditions des codes S-CSO^{2}C-WS ce qui nous permet de créer une nouvelle connexion α_3 égale à $(\alpha_2 + 1)$. Maintenant, l'ensemble $\mathcal{A}$ contient les trois éléments suivants {0, 1, 2}. Cet ensemble ne répond pas aux critères, et par conséquent la valeur α_3 est augmentée de 1

générant l'ensemble {0, 1, 3}. Cet ensemble ne répond pas non plus aux critères, donc nous augmentons de 1 la valeur de α_3 pour obtenir l'ensemble $\mathcal{A}$={0, 1, 4}. Cet ensemble répond aux critères donc une solution est trouvée. L'objectif de la recherche étant de minimiser la longueur des codes, il est évident qu'en possédant cette solution nous allons cibler la recherche vers des codes de longueur inférieure à 4. Par conséquent, la métrique α^{max} est mise à jour et devient égale à 4.

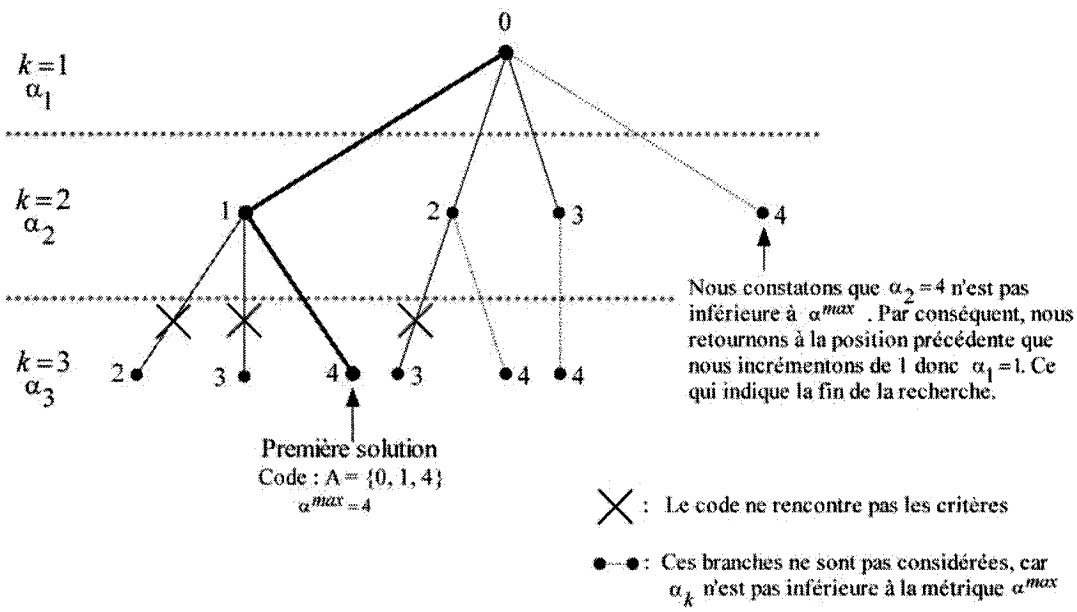

Figure 3.3: Parcours effectué dans l'arbre de recherche associé à l'exemple 3.2

Lorsqu'une solution est trouvée, nous retournons à la position précédente, dans notre cas α_2, que nous incrémentons de 1 pour obtenir α_2 égale à 2. Cette opération à pour but de changer de branche dans l'arbre de recherche. Dans un premier temps, nous vérifions que la valeur de la nouvelle connexion est inférieure à la métrique. Par la suite, nous vérifions que l'ensemble {0, 2} répond aux critères. Cet ensemble vérifie les conditions ce qui nous permet de générer la con-

nexion suivante α_3 égale à $(\alpha_2 + 1) = 3$. Cette connexion est inférieure à la métrique donc nous vérifions si l'ensemble {0, 2, 3} répond aux critères. Cet ensemble ne répond pas aux critères des codes simplifiés alors nous augmentons la valeur de α_3 à $(\alpha_3 + 1) = 4$. Nous constatons que cette valeur n'est pas inférieure à la métrique α^{max} donc nous ne testons pas le code et nous retournons à la connexion précédente que nous incrémentons de une unité $\alpha_2 = 3$. Cette valeur est inférieure à α^{max} et l'ensemble {0, 3} répond aux conditions. Nous créons une nouvelle connexion α_3 égale à $(\alpha_2+1)=4$. Cette connexion n'est pas inférieure à α^{max} donc nous revenons à la connexion précédente α_2 que nous incrémentons de 1. Nous constatons que $(\alpha_2 + 1) = 4$ n'est pas inférieur à la métrique donc nous retournons à la connexion précédente α_1 que nous incrémentons de 1 donc $\alpha_1 = 1$ ce qui indique la fin de la recherche dans l'arbre, car tous les codes débutent par la valeur zéro $(\alpha_1 = 0)$. Donc, le plus petit code S-CSO^{2}C-WS de dimension 3 est donné par l'ensemble $\mathcal{A} = \{0, 1, 4\}$.

Suite aux recherches effectuées, nous avons constaté que l'algorithme n'est plus efficace lorsque la dimension des codes est supérieure à 8. À titre indicatif, la recherche a duré plus de deux semaines pour parcourir entièrement l'arbre de recherche pour $J = 8$. Le tableau 3.2 présente les temps requis pour parcourir l'arbre de recherche pour trouver les plus petits codes S-CSO^{2}C-WS.

Tableau 3.2: Temps nécessaire pour parcourir l'arbre de recherche, $4 \leq J \leq 8$

J	4	5	6	7	8
Temps (secondes)	< 1	< 1	11	632	976209

- **Recherche aléatoire**

Pour pallier à l'inefficacité en temps de l'algorithme de recherche exhaustif, nous avons développé une heuristique basée sur une recherche aléatoire. Ce type d'algorithme permet l'obtention de codes de dimensions élevées, $J > 8$, à l'intérieur de délais raisonnables. Cependant, bien que les codes trouvés soient de bons codes, rien ne garantit qu'ils soient de longueur minimum.

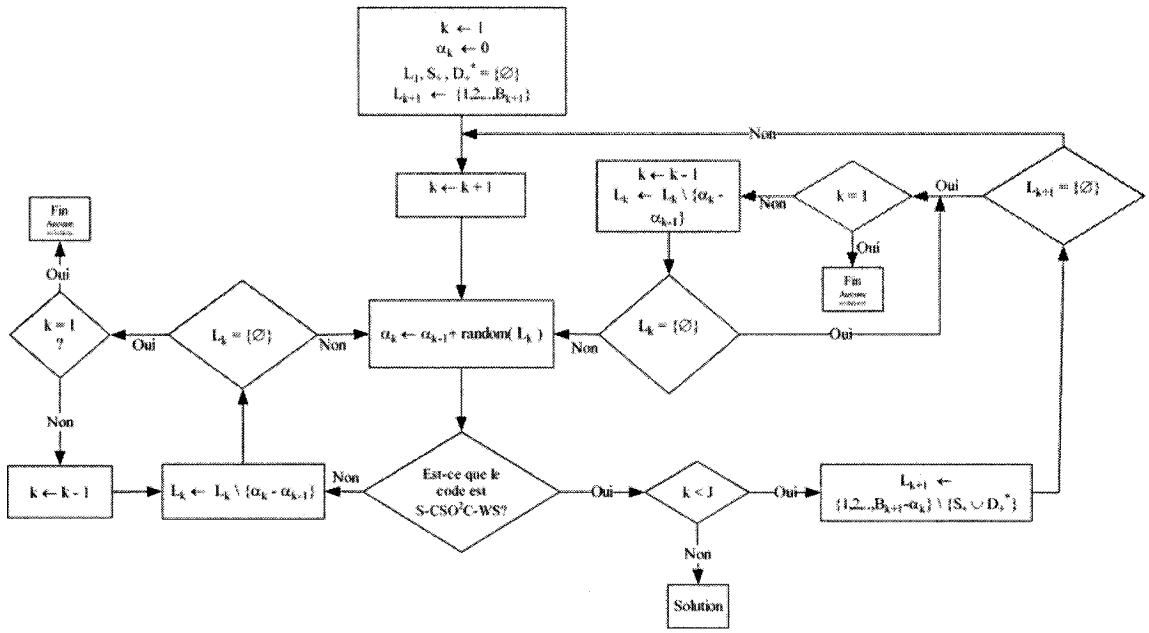

Figure 3.4: Diagramme représentant l'algorithme de recherche aléatoire

Contrairement à l'algorithme de recherche exhaustif, celui-ci nécessite la connaissance de la longueur d'un code du même ordre. Ce paramètre α_J^{max}, qui correspond à la longueur du plus petit code CSO^{2}C-WS connu, permet le calcul des bornes B_k sur l'ensemble des connexions $\{\alpha_k\}$ du code. Nous créons des listes de différences simples possibles, car les connexions seront générées à partir de différences simples.

Par exemple, la k-ième connexion est créée de la façon suivante :

$$\alpha_k = \alpha_{k-1} + s_{k,k-1} \tag{3.10}$$

où $s_{k,k-1}$ représente la différence simple tirée de façon aléatoire dans la liste L_k et où α_{k-1} est la valeur associée à la connexion précédente. La liste L_k est obtenue selon la relation suivante :

$$L_k = \{1, 2, 3, \ldots, (B_k - \alpha_{k-1})\} \setminus S_+ \cup D_+^* \tag{3.11}$$

où B_k représente la borne supérieure associée à la k-ième connexions et les ensembles S_+ et D_+^* représentent les différences simples et doubles positives (sans les inévitables) associées aux $(k-1)$ premières connexions du code.

Il s'en suit que si la connexion à créer α_k possède la borne supérieure B_k alors la valeur maximale que peut prendre la liste L_k (donc la différence simple $s_{k,k-1}$) est $B_k - \alpha_{k-1}$. On en déduit que $s_{k,k-1}$ est compris entre 1 et $B_k - \alpha_{k-1}$. De ces valeurs possibles, nous pouvons éliminer les différences simples et doubles engendrées par le code constitué des $(k-1)$ premières connexions, car la définition des codes simplifiés exige que les différences simples et doubles soient différentes. Autrement dit, la différence simple $s_{k,k-1}$ ne peut pas faire partie des ensembles S_+ et D_+^* engendrés par les connexions connues. Si la liste est vide alors il n'est pas possible de créer un code à partir des $(k-1)$ premières connexions. Dans ce cas, l'algorithme retourne en arrière pour éliminer la connexion α_{k-1} de la liste L_{k-1} et tire une nouvelle valeur dans cette liste. Lorsque la connexion α_k est créée le code composé de k connexions est testé. S'il répond à la définition des codes simplifiés l'algorithme poursuit sa recherche pour une nouvelle valeur α_{k+1}. Dans le cas contraire, si le code ne répond pas à la définition alors la valeur tirée $s_{k,k-1}$ est retirée de la liste L_k et une nouvelle valeur est tirée. Si aucun code n'est trouvé après un certain temps alors la recherche arrête.

La création des bornes sur les connexions est assez simple. Les bornes sont basées sur la valeur maximale α_J^{max} que peut prendre le code simplifié. Donc, la dernière connexion du code α_J est inférieure à la valeur maximale $\alpha_J < \alpha_J^{max}$ de plus, nous imposons à la première connexion α_1 la valeur zéro. Par la suite, nous utilisons la règle suivante :

$$\alpha_{\lceil \frac{J}{2^i} \rceil} < \frac{(\alpha_J^{max} - 1)}{2^i} = B_{\lceil \frac{J}{2^i} \rceil} \tag{3.12}$$

pour $i = 0, \ldots, \lceil log_2(J) \rceil - 1$.

Exemple 3.3- Cet exemple montre le parcours effectué par l'algorithme de recherche aléatoire pour trouver un code S-CSO^{2}C-WS de dimension $J = 3$, $\mathcal{A} = \{\alpha_1, \alpha_2, \alpha_3\}$.

Sachant que le plus petit code CSO^{2}C-WS de dimension 3 est donné par l'ensemble {0, 1, 5}, le paramètre α_3^{max} choisi sera égale à 4 et la borne sur la dernière connexion sera $B_3 = 4$. En utilisant la relation (3.12) nous pouvons obtenir la valeur de la borne pour la deuxième connexion et $\alpha_2 < B_2 = 2$. Par conséquent la liste L_2 relative à la connexion α_2 ne peut que contenir la valeur égale à 1, car la connexion doit être inférieure à 2. On tire aléatoirement dans la liste une valeur (ici il n'y a qu'une valeur) donc $\alpha_2 = (\alpha_1 + 1)$. L'ensemble formé est {0, 1} et il répond aux critères des codes simplifiés. Les ensembles des différences simples et doubles relatifs à ce code sont $S_+ = \{1\}$ et $D_+^* = \{2\}$. La liste $L_3 = \{1, \ldots, (B_3 - \alpha_2) = 3\} \setminus \{1, 2\} = \{3\}$. Nous tirons aléatoirement une valeur et $\alpha_3 = \alpha_2 + 3 = 4$. L'ensemble {0, 1, 4} est créé et testé. Ce code répond aux conditions pour un code S-CSO^{2}C-WS, donc une solution est trouvée.

Les tableaux 3.3, 3.4 et 3.5 présentent les meilleurs ensembles de connexions obtenus pour différentes valeurs de J avec les deux algorithmes présentés dans cette section. Nous avons intégré dans ces tableaux les meilleurs codes CSO^2C-WS (les plus petits au sens de la longueur), $\delta = 0$, pour faciliter la comparaison entre la longueur des codes S-CSO^2C-WS et CSO^2C-WS.

3.5.2 Liste de codes S-CSO^{2}C-WS

Tableau 3.3: Ensembles des meilleures connexions $\alpha_k \in \mathcal{A}$ pour les codes S-CSO^{2}C-WS, $5 \leq J \leq 10$

J	δ	Ensemble des connexions $\{\alpha_k\}$
5	0	{0, 1, 24, 37, 41}
	0.2181	{0, 1, 10, 25, 32}
	0.3636	{0, 1, 15, 20, 23}†
	0.3818	{0, 1, 15, 18, 23}†
6	0	{0, 1, 17, 70, 95, 100}
	0.2333	{0, 35, 47, 67, 69, 76}
	0.4333	{0, 2, 11, 26, 42, 45}†
7	0	{0, 1, 53, 128, 207, 216, 222}
	0.2294	{0, 48, 95, 121, 137, 154, 156}
	0.3679	{0, 3, 5, 33, 73, 82, 95}
	0.4286	{0, 1, 7, 50, 59, 78, 82}†
	0.4329	{0, 2, 23, 45, 72, 79, 82}†
	0.4416	{0, 4, 23, 32, 75, 76, 82}†
8	0	{0, 43, 139, 322, 422, 430, 441, 459}
	0.2241	{0, 40, 157, 200, 249, 253, 275, 287}
	0.4261	{0, 4, 19, 44, 95, 130, 144, 147}
	0.4433	{0, 1, 6, 45, 100, 119, 127, 142}
	0.4828	{0, 9, 22, 55, 95, 124, 127, 129}†
9	0	{0, 9, 21, 395, 584, 767, 871, 899, 912}
	0.2357	{0, 41, 196, 215, 346, 349, 385, 446, 495}
	0.3889	{0, 17, 28, 78, 190, 216, 256, 263, 264}
	0.4895	{0, 1, 17, 26, 127, 138, 185, 204, 208}
10	0	{0, 29, 40, 43,1020,1328,1495,1606,1696,1698}
	0.1363	{0, 128, 261, 410, 534, 698, 743, 891, 1038, 1190}
	0.3575	{0, 2, 10, 31, 103, 219, 316, 370, 447, 454}
	0.4106	{0, 95, 113, 145, 291, 346, 400, 424, 443, 453}
	0.4638	{0, 1, 5, 12, 61, 140, 251, 294, 327, 352}
	0.4917	{0, 1, 87, 93, 226, 262, 296, 316, 327, 340}

† Ces codes simplifiés, obtenus à l'aide de l'algorithme de recherche exhaustif, sont ceux ayant la plus petite longueur possible ainsi ils sont optimaux au sens de la longueur.

Tableau 3.4: Ensembles des meilleures connexions $\alpha_k \in \mathcal{A}$ pour les codes S-CSO^{2}C-WS, $11 \leq J \leq 15$

J	δ	Ensemble des connexions $\{\alpha_k\}$
11	0	{0, 8, 18, 25, 73, 150, 346, 839, 1900, 2955, 3467}
	0.0935	{0, 290, 487, 633, 890, 1153, 1243, 1542, 1858, 2117, 2239}
	0.1786	{0, 10, 211, 441, 541, 808, 1067, 1120, 1256, 1317, 1582}
	0.3071	{0, 117, 130, 174, 192, 443, 643, 717, 801, 808, 923}
	0.4539	{0, 1, 5, 12, 32, 61, 199, 350, 434, 480, 588 }
12	0	{0, 48, 212, 1014, 1381, 2217, 4198, 4373, 4766, 4885, 4914, 5173}
	0.1922	{0, 6, 79, 280, 482, 693, 972, 1108, 1483, 1575, 1998, 2035}
	0.2420	{0, 39, 56, 249, 325, 489, 816, 1183, 1279, 1368, 1627, 1631}
	0.3311	{0, 14, 16, 60, 124, 319, 481, 708, 906, 1042, 1048, 1061}
	0.4632	{0 1 5 12 32 61 107 271 411 584 707 894}
13	0	{0, 39, 41, 44, 99, 230, 485, 785, 1693, 2913, 5319, 7183, 9252}
	0.1418	{0, 480, 818, 826, 1458, 1562, 1869, 2519, 2667, 3260, 3872, 4154, 4232}
	0.2798	{0, 228, 242, 297, 523, 586, 785, 796, 1001, 1635, 1738, 2052, 2356}
	0.4193	{0, 2, 12, 144, 190, 207, 633, 747, 974, 1052, 1111, 1214, 1217 }
14	0	{0, 6, 10, 19, 59, 160, 384, 862, 1843, 2903, 5176, 7943, 12467, 13774}
	0.1369	{0, 79, 185, 369, 658, 693, 1412, 2398, 3307, 3655, 3990, 4411, 5363, 6202}
	0.2494	{0, 100, 132, 307, 656, 732, 1220, 1667, 2179, 2198, 2541, 2624, 2738, 3579}
	0.4269	{0, 1, 5, 12, 32, 61, 107, 230, 355, 514, 919, 1188, 1660, 1967 }
15	0	{0, 26, 31, 68, 71, 122, 438, 445, 1539, 3797, 3969, 6570, 13785, 16503, 18548}
	0.1035	{0, 358, 577, 666, 1693, 2807, 3737, 4954, 6087, 7347, 7994, 8478, 9992, 10134, 10469}
	0.1655	{0, 137, 929, 1143, 1528, 1875, 2140, 3082, 4207, 4384, 4896, 5284, 6388, 6778, 7734}
	0.4253	{0, 1, 5, 12, 32, 61, 107, 230, 355, 514, 824, 1424, 1726, 2384, 2653 }

Tableau 3.5: Ensembles des meilleures connexions $\alpha_k \in \mathcal{A}$ pour les codes S-CSO^{2}C-WS, $16 \leq J \leq 20$

J	δ	Ensemble des connexions $\{\alpha_k\}$
16	0	{0, 5, 19, 27, 75, 170, 406, 867, 1505, 2715, 5267, 8828, 13569, 19449, 28349, 34908}
	0.1609	{0, 2851, 3494, 5311, 6915, 8488, 9052, 9337, 9918, 10335, 10453, 10472, 10576, 10597, 10706, 10709}
	0.2218	{0, 1426, 2101, 2381, 4323, 5872, 6114, 6570, 6699, 6782, 6912, 7174, 7319, 7456, 7522, 7549}
	0.4313	{ 0, 1, 5, 12, 32, 61, 107, 230, 355, 514, 824, 1255, 1727, 2254, 3262, 3532}
17	0	{0, 4, 7, 30, 98, 168, 391, 885, 1846, 2988, 5442, 9382, 13426, 21519, 35643, 45366, 50071}
	0.0435	{0, 305, 5995, 11263, 12154, 24258, 24625, 26607, 30262, 34989, 35809, 36967, 37543, 38282, 38341, 38454, 38909}
	0.1957	{0, 5172, 7282, 7607, 8997, 9805, 10511, 10939, 11096, 12354, 12580, 13171, 13535, 13571, 13759, 13893, 13953}
	0.4246	{0, 1, 12, 39, 47, 108, 137, 220, 376, 589, 873, 1244, 1641, 2216, 3151, 3975, 4978}
18	0	{0, 2, 10, 21, 81, 167, 427, 797, 1703, 3043, 4991, 9073, 15783, 20973, 35336, 42994, 61796, 71858}
	0.1703	{0, 543, 3755, 3902, 4754, 7558, 10867, 12158, 12180, 13069, 13698, 13728, 14166, 14554, 14636, 14654, 14792, 14834}
	0.2102	{0, 295, 1613, 1918, 5042, 5339, 7987, 8028, 8815, 9519, 10633, 10995, 11424, 11484, 11590, 11599, 11602, 11800}
	0.4002	{0, 1, 12, 39, 47, 111, 178, 219, 378, 629, 831, 1315, 1914, 2648, 3331, 4420, 5507, 6905}
19	0	{0, 5, 13, 51, 57, 175, 472, 968, 1579, 2771, 5680, 9348, 13795, 20573, 33417, 51727, 73040, 96889, 107528}
	0.2945	{0, 491, 866, 1135, 1478, 1733, 2143, 2504, 2940, 2964, 2983, 3681, 4641, 5512, 7004, 8358, 10331, 10726, 13116}
	0.4053	{0, 1, 12, 39, 47, 111, 178, 219, 378, 629, 831, 1315, 1914, 2648, 3331, 4420, 5507, 6905, 8748}
20	0	{0, 23, 25, 51, 87, 193, 457, 996, 1698, 3225, 5423, 9379, 13147, 21753, 34552, 50375, 70040, 95517, 124463, 148787}
	0.3806	{0, 1, 182, 268, 396, 627, 935, 1066, 1225, 1549, 2058, 2279, 2908, 4210, 5575, 6334, 7539, 9169, 9429, 10768}
	0.3923	{0, 1, 280, 404, 583, 622, 867, 1048, 1414, 1883, 2452, 2906, 2923, 3860, 5132, 6693, 7384, 9490, 9701, 9749}

3.5.3 Variation de la longueur minimale des codes simplifiés S-CSO^{2}C-WS et CSO^{2}C-WS

La figure 3.5 présente la comparaison des longueurs des codes obtenus en fonction de leur dimension J. Nous observons que plus J est élevé, plus la réduction de la longueur des codes est grande par rapport à celle des codes CSO^{2}C-WS les plus courts que l'on connaisse jusqu'à présent.

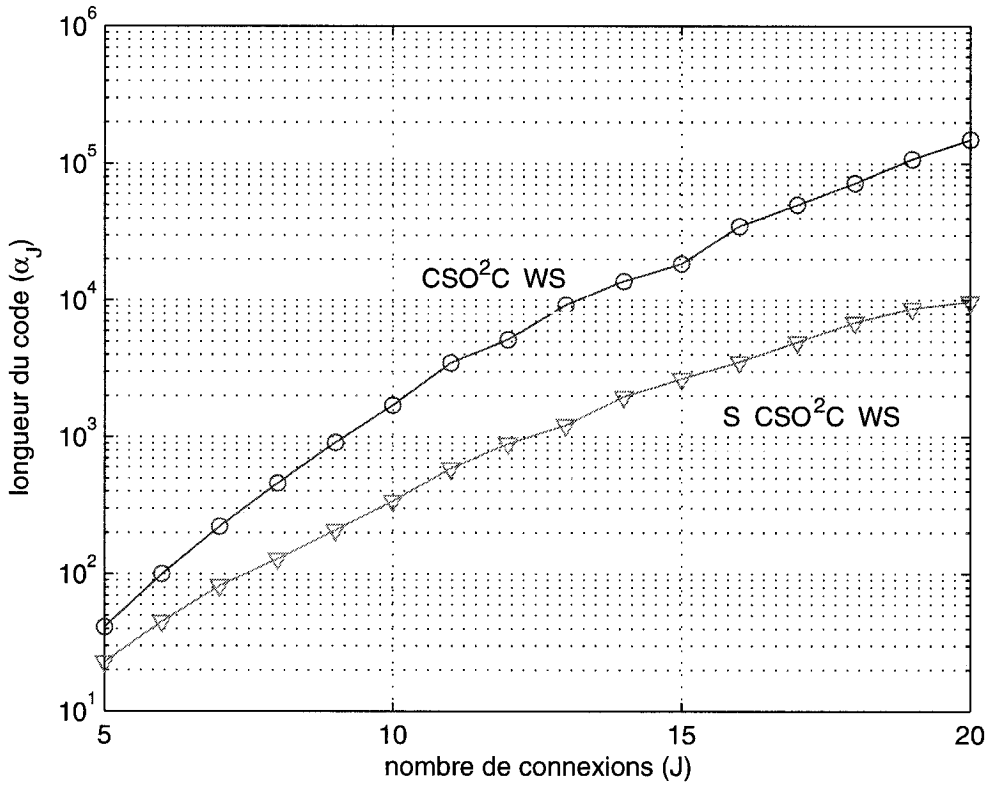

Figure 3.5: Variation de la longueur des codes CSO^{2}C-WS et S-CSO^{2}C-WS, $5 \leq J \leq 20$

Nous pouvons aussi représenter ces résultats en exprimant la réduction maximale de longueur obtenue en fonction de J. La réduction de la longueur entre deux codes s'exprime par :

$$\rho_J = 1 - \frac{\alpha^S}{\alpha^C} \tag{3.13}$$

où α^S et α^C représentent les plus petites longueurs connues de dimension J pour les codes S-CSO^{2}C-WS et CSO^{2}C-WS respectivement. Par exemple, lorsque $J = 10$, la plus petite longueur pour un code simplifié est $\alpha_{10}^S = 340$ et la plus petite longueur pour un code CSO^{2}C-WS est $\alpha_{10}^C = 1698$ donc, $\rho_{10} = 0.7998$. Nous remarquons aussi que la figure 3.6 peut représenter la réduction de la latence totale entre les codes S-CSO^{2}C-WS et CSO^{2}C-WS lorsque le nombre d'itérations et la dimension entre les codes sont les mêmes.

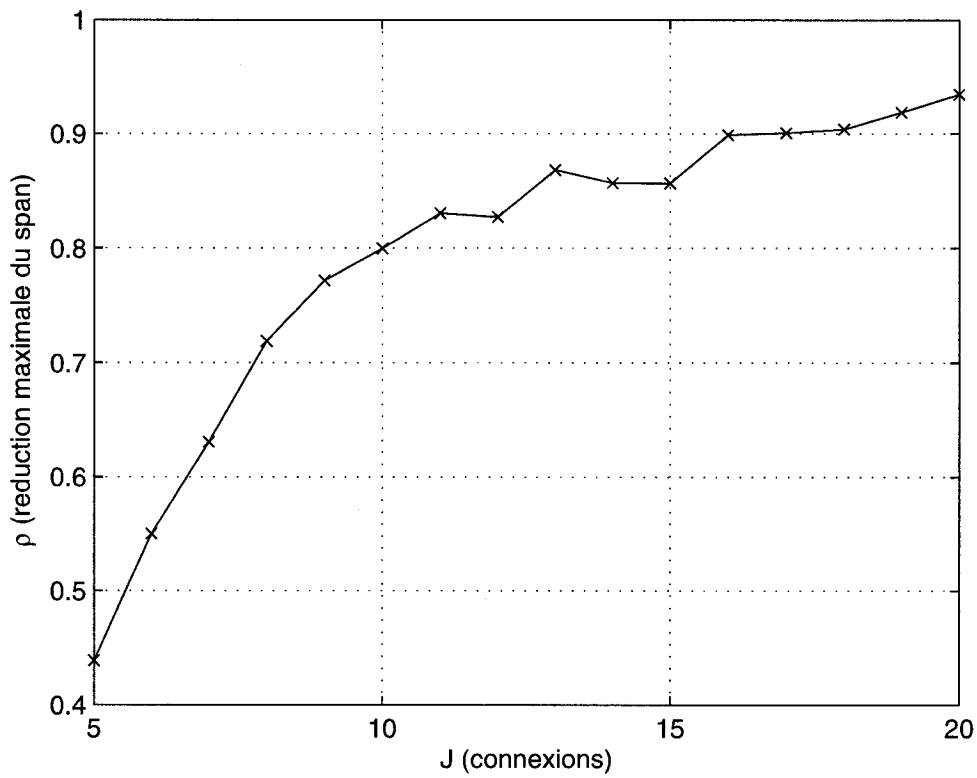

Figure 3.6: Réduction de la longueur entre les plus petits codes CSO^{2}C-WS et S-CSO^{2}C-WS, $5 \leq J \leq 20$, $R = \frac{1}{2}$

- **Borne inférieure sur la longueur des codes S-CSO^{2}C-WS**

Nous pouvons définir une borne inférieure, α_J^*, sur la longueur minimum des codes simplifiés avec la relation suivante [6] .

$$\alpha_J^* = \lceil \frac{N_s + (1-\delta)N_d}{2} \rceil \tag{3.14}$$

Cette borne inférieure peut servir de repère pour situer la longueur minimale des codes trouvés aux tableaux 3.3 et 3.4. La figure 3.7 montre que plus le facteur de simplification est élevé plus la longueur des codes simplifiés tend à s'approcher de la borne inférieure.

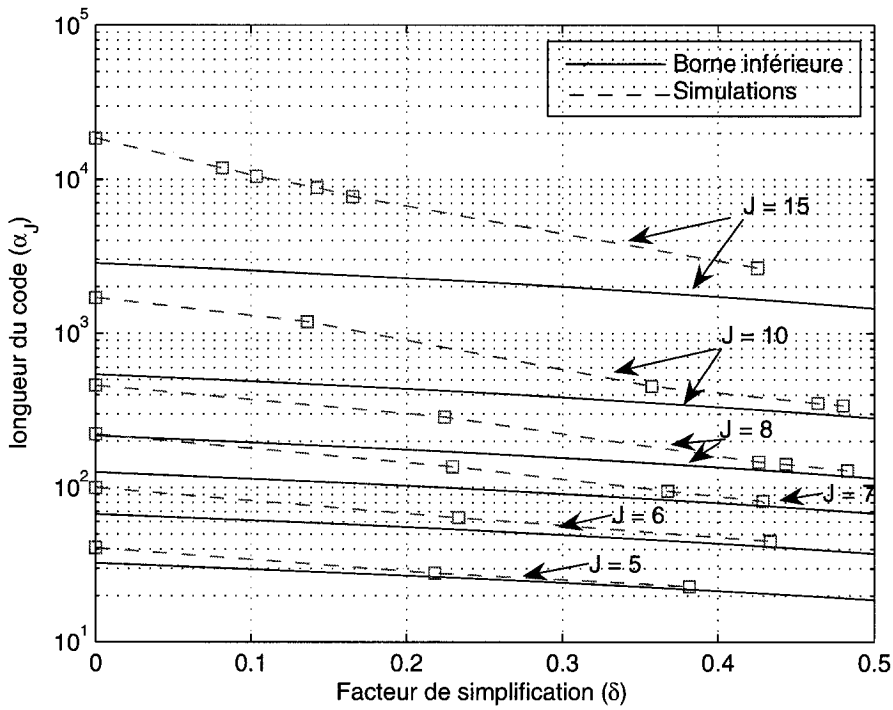

Figure 3.7: Comparaison entre la borne inférieure sur la longueur des codes et les valeurs obtenues par la recherche pour $J = 5, 6, 7, 8, 10, 15$, $R = \frac{1}{2}$

[6] Voir l'annexe I pour la justification de cette borne

Remarquons que pour tous les codes simplifiés trouvés, la tendance indique que le facteur de simplification δ est inférieur à $\frac{1}{2}$ ce qui est assez loin de la borne supérieure exprimée par (3.7). Ceci dit, existe-t-il des codes simpliés ayant $\delta > \frac{1}{2}$? Cette question reste un problème ouvert. Nous pouvons dégager un compromis important de cette figure en remarquant que les codes simplifiés ayant la plus petite longueur sont obtenus en introduisant plus d'éléments correlés dans les équations de parité $\psi_{i,k}^{(2)}$ données par (3.4). Cependant, quel est le prix à payer en terme de performances d'erreur lorsque nous utilisons les codes fortement simplifiés ?

3.5.4 Analyse des performances d'erreur des codes S-CSO^{2}C-WS

Dans cette section, nous exposons les performances d'erreur[7] du décodeur itératif à seuil lorsque les codes simplifiés S-CSO^{2}C-WS sont utilisés. L'étude des résultats de simulation à l'ordinateur, nous a permis de mesurer l'influence de la simplification des codes sur : la convergence de l'algorithme de décodage, les probabilités d'erreur et le délai total de décodage.

- **Convergence de l'algorithme de décodage à seuil itératif pour les codes S-CSO^{2}C-WS**

La figure 3.8 présente les performances d'erreur obtenues pour le code le plus simplifié de dimension 10 du tableau 3.3. Nous constatons que le code simplifié offre une amélioration des performances d'erreur d'une itération à l'autre tout comme le code CSO^{2}C-WS. Cependant, nous pouvons remarquer une dégradation du gain codage entre ces deux codes lorsque le canal est fortement bruité. Cette dégradation s'accentue d'une itération à l'autre, car l'introduction des différences doubles égales amplifie la propagation des erreurs réduisant ainsi les performances du décodeur.

[7] À l'annexe V nous présentons plusieurs résultats de simulations complémentaires à ce chapitre

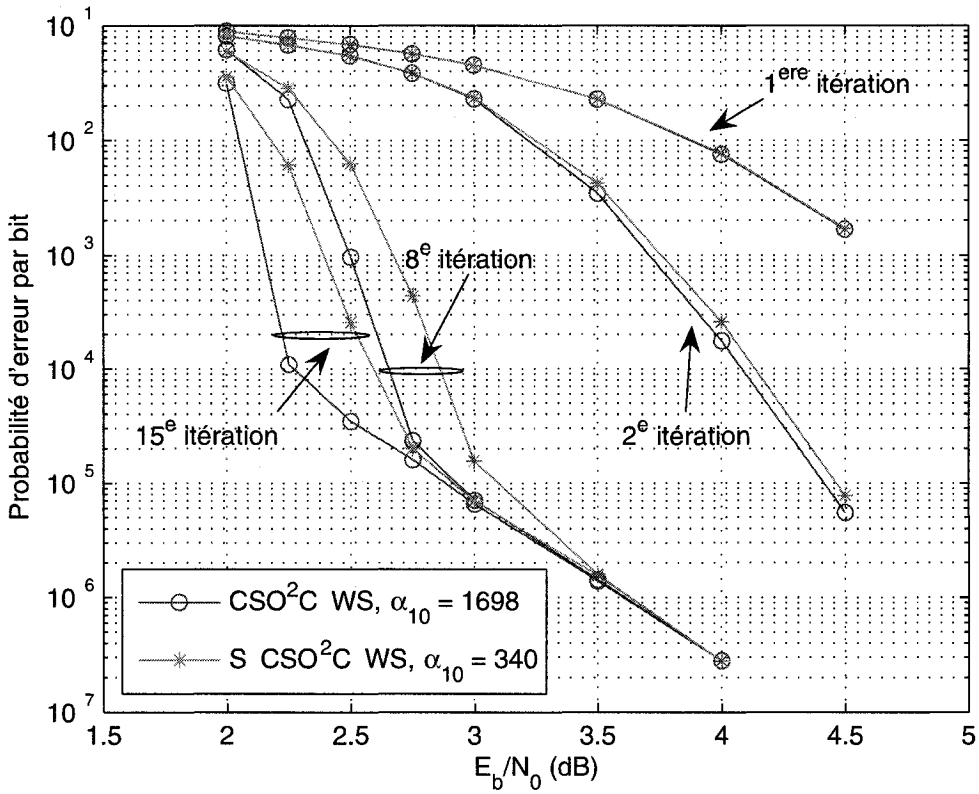

Figure 3.8: Performance d'erreur pour le code simplifié de dimension $J = 10$, $\{0, 1, 87, 93, 226, 262, 296, 316, 327, 340\}$, $\delta = 0.4801$, $a^* = 0.2$, $R = \frac{1}{2}$

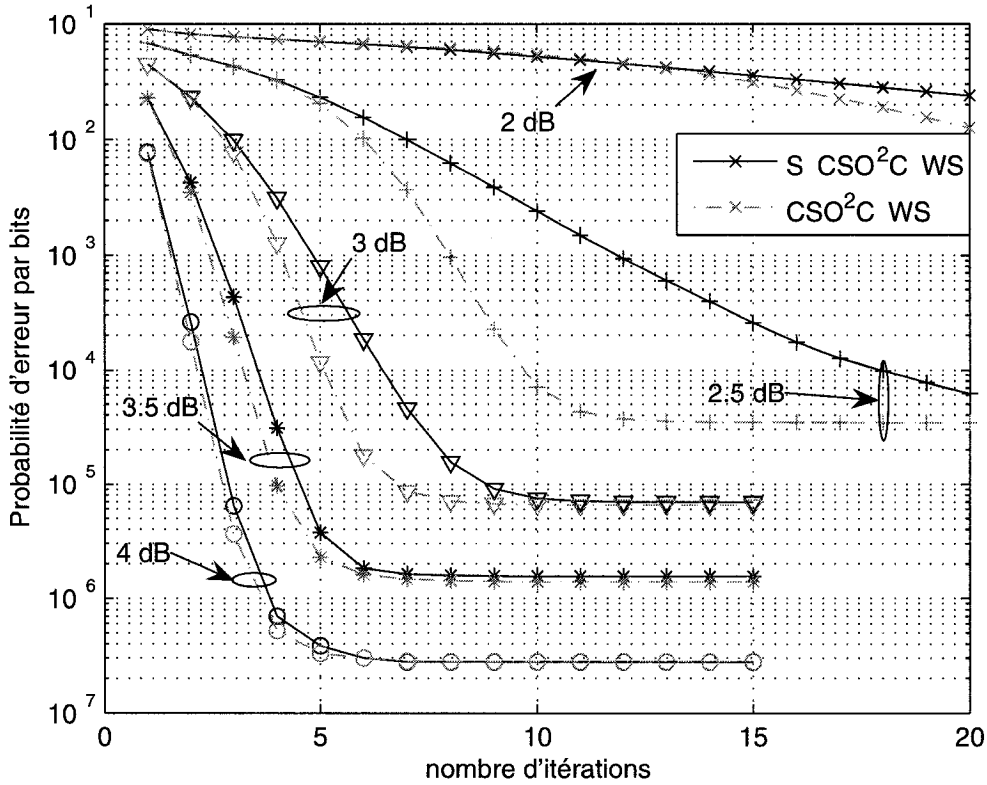

Figure 3.9: Comparaison entre la convergence de l'algorithme itératif à seuil pour le code S-CSO^{2}C-WS $J = 10$, $\{0, 1, 87, 93, 226, 262, 296, 316, 327, 340\}$, $\delta = 0.48$ et le code CSO^{2}C-WS $J = 10$, $\{0, 29, 40, 43, 1020, 1328, 1495, 1606, 1696, 1698\}$, $a^* = 0.2$, $R = \frac{1}{2}$

La figure 3.9 montre la capacité de correction du décodeur en fonction des itérations effectuées par celui-ci pour les deux codes présentés à la figure 3.8. D'une part, nous remarquons que pour différentes valeurs du rapport E_b/N_0 le décodeur à seuil atteint un seuil à partir duquel l'ajout d'itérations n'améliore plus la capacité de correction. Par exemple, si $E_b/N_0 = 3$ dB peu importe le type de code utilisé le décodeur à seuil itératif ne permet pas l'obtention d'une probabilité d'erreur inférieure à 7 x 10^{-6}. Pour cet exemple, le code simplifié atteint la capacité maximale de correction en 10 itérations soit deux de plus que le code CSO^{2}C-WS. Bien que le nombre d'itérations soit plus élevé lorsque nous utilisons le code simplifié, il n'en reste pas moins que la

latence totale demeure inférieure à celle du code CSO^{2}C-WS. De plus, nous pouvons aussi remarquer que le nombre d'itérations supplémentaires à effectuer, en utilisant le code simplifié, s'estompe à mesure que le rapport signal sur bruit augmente.

- **Performances d'erreur des codes simplifiés en fonction du rapport de simplification**

La figure 3.10 présente les performances d'erreur des codes simplifiés de dimension 10 du tableau 3.3 à la huitième itération. On constate que le code le plus simplifié adopte le même comportement que le code CSO^{2}C-WS lorsque $E_b/N_0 > 3.5$ dB. Nous avons pu observer, via les simulations, que même les codes les plus simplifiés convergent en terme de probabilité d'erreur vers les performances des codes CSO^{2}C-WS.

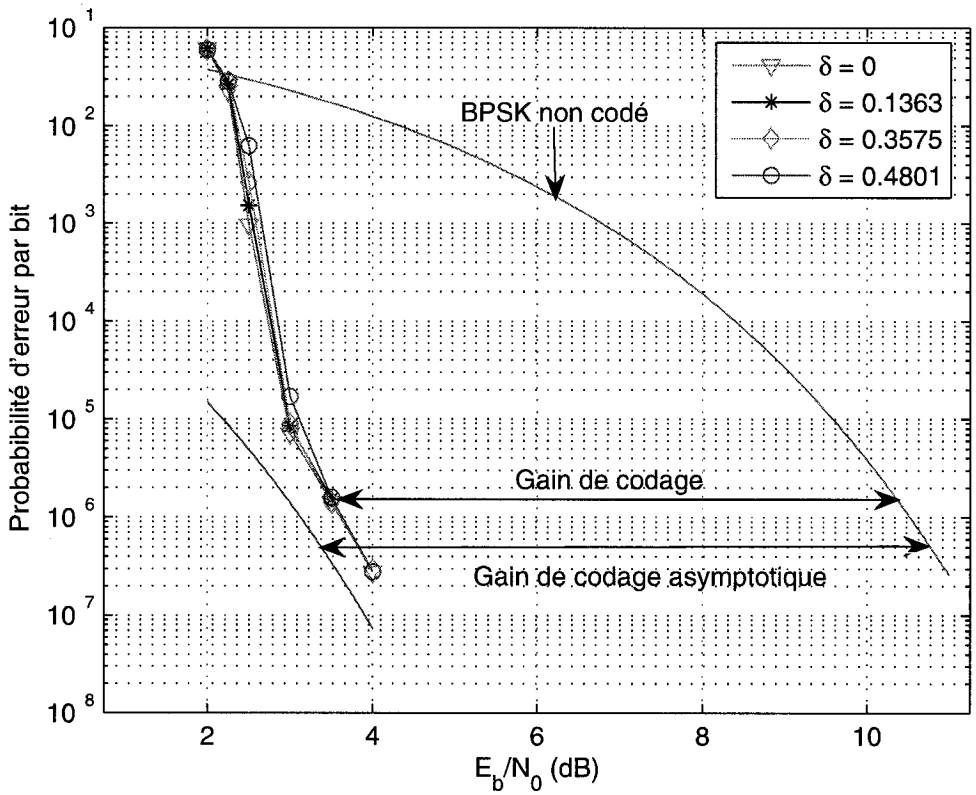

Figure 3.10: Comparaison entre les performances d'erreur des codes simplifiés S-CSO^{2}C-WS à la 8^e itération, $J = 10$, $a^* = 0.2$, $R = \frac{1}{2}$

Il est intéressant de constater avec cet exemple, que quelque soit le code utilisé, pour une probabilité d'erreur égale à 10^{-6}, il est possible d'obtenir un gain de codage d'environ 6.9 dB par rapport au cas BPSK non codé. À performances égales, le code le plus simplifié permet d'économiser 80 % de la latence totale par rapport au code CSO^{2}C-WS. Le compromis entre la réduction du délai de décodage total et les performances est donc fort avantageux. La section suivante analyse de plus près ce compromis.

- **Performances d'erreur des codes simplifiés en fonction de leur latence totale**

Comme nous l'avons vu précédemment, le fort délai de décodage induit par les codes CSO^{2}C-WS est une problématique pouvant être résolue en utilisant les codes simplifiés. La figure (3.11) compare la variation des probabilités d'erreur des codes les plus simplifiés en fonction de leur latence totale de décodage avec celles des codes CSO^{2}C-WS.

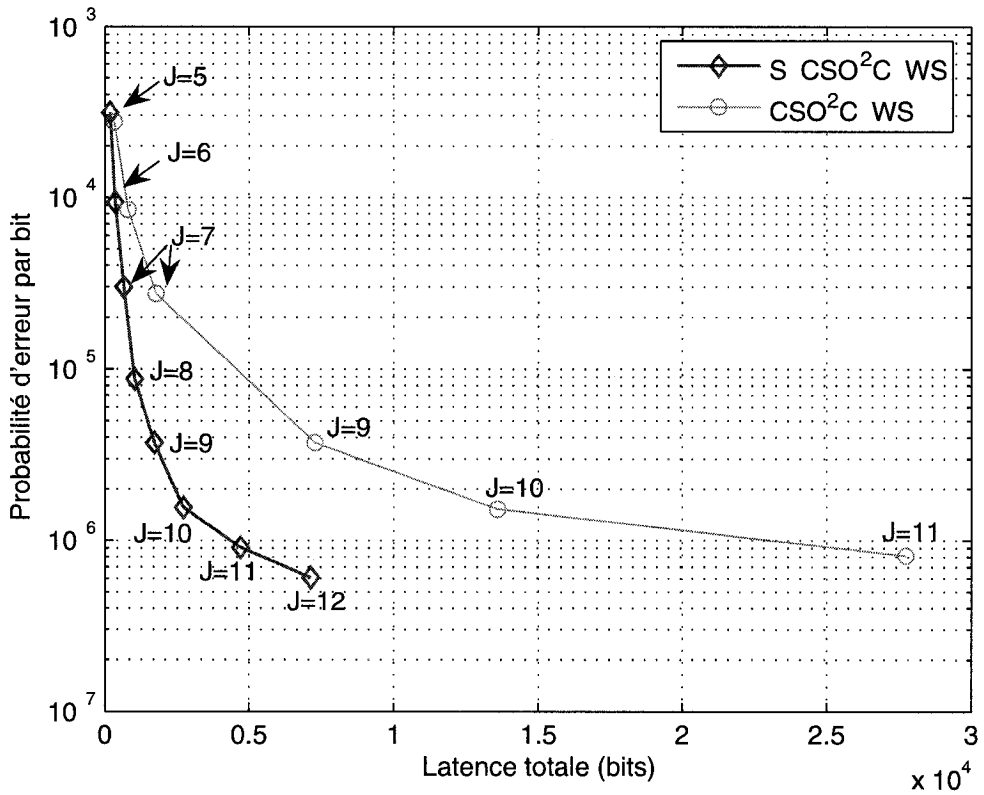

Figure 3.11: Comparaison entre les performances d'erreur des codes les plus simplifiés S-CSO^{2}C-WS et CSO^{2}C-WS en fonction de la latence totale, 8^e itération, $R = \frac{1}{2}$, $E_b/N_0 = 3.5$ dB

D'une part, il est intéressant de constater que les performances d'erreur pour les codes les plus simplifiés sont sensiblement les mêmes que celles obtenues avec les

codes CSO^{2}C-WS lorsque la dimension des codes est la même . D'autre part, il est clair qu'en fixant la latence totale au décodage, que les codes simplifiés deviennent largement avantageux. Par exemple, le meilleur code CSO^{2}C-WS de dimension $J = 9$ induit un délai de décodage de 7296 bits contre 7152 pour le code le plus simplifié de dimension $J = 12$ lorsqu'on effectue huit itérations. Les performances d'erreur de ces deux codes sont représentées à la figure 3.12.

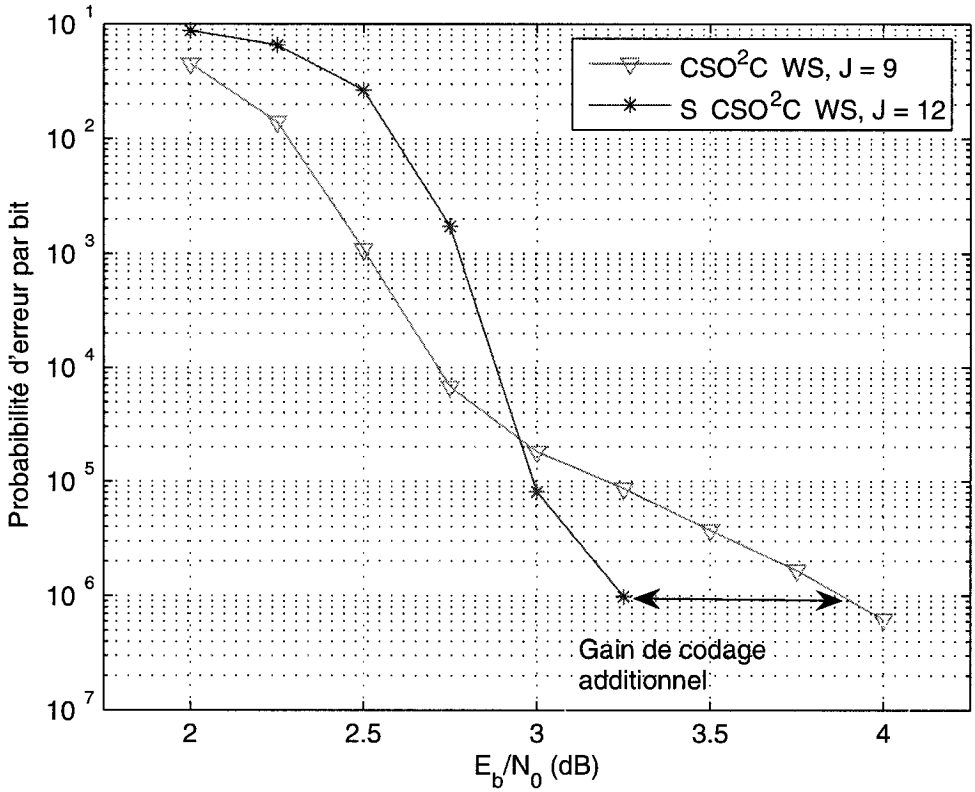

Figure 3.12: Comparaison entre les performances d'erreur des codes S-CSO^{2}C-WS (pire cas) et CSO^{2}C-WS , 8^e itération

Or, à fort rapport signal sur bruit, pour un délai de décodage sensiblement identique, on obtient de meilleurs performances d'erreur avec le code simplifié. Pour cet exemple, lorsque la probabilité d'erreur par bit est fixée à 10^{-6} il est possible d'obtenir un gain de codage additionnel d'environ 0.6 dB en utilisant le code simplifié $J = 12$.

3.6 Conclusion

Les résultats obtenus dans ce chapitre, nous permettent de constater que les codes simplifiés de taux de codage 1/2 se révèlent être des codes avec lesquels il est possible d'obtenir des performances d'erreur semblables à celles obtenues avec les codes CSO^{2}C-WS. Par conséquent, les performances d'erreur des codes simplifiés s'améliorent pour des valeurs de J croissantes et varient très peu en fonction du facteur de simplification. Avantageusement, ils offrent une réduction supérieure à 80 % de la latence totale, lorsque J est supérieur à 10, et permettent une réduction considérable des composants physiques par rapport aux codes CSO^{2}C-WS. Pour le pire des cas, nous avons constaté une dégradation négligeable du gain de codage par rapport aux codes CSO^{2}C-WS à haut rapport signal sur bruit.

De plus, nous avons remarqué que les codes simplifiés peuvent offrir de meilleures performances d'erreur que les codes CSO^{2}C-WS pour un délai de décodage fixe. Cette caractéristique provient du fait qu'il est possible de réduire fortement la longueur des codes en relâchant certaines des conditions sur les propriétés de double orthogonalité des codes CSO^{2}C-WS.

CHAPITRE 4

CODES CONVOLUTIONNELS DOUBLEMENT ORTHOGONAUX SIMPLIFIÉS ET PERFORÉS AU SENS LARGE (S-PCSO^{2}C-WS)

4.1 Introduction

Dans ce chapitre, nous analysons l'influence de la perforation des codes convolutionnels systématiques (SCC) de taux de codage $R = \frac{1}{2}$ sur les performances d'erreur du décodeur itératif à seuil présenté au chapitre précédent. Toujours afin de minimiser la latence totale lors du décodage, nous réduirons les contraintes imposées aux codes perforés dans le but de diminuer leurs longueurs. Nous rechercherons alors des codes SCC de taux de codage $R = \frac{1}{2}$ ne répondant pas à la définition des codes simplifiés S-CSO^{2}C-WS au départ, mais une fois que ces codes seront perforés à un taux de codage $R = \frac{b}{b+1}$, ils acquièront toutes les propriétés des codes doublement orthogonaux simplifiés au sens large. Les codes simplifiés trouvés seront ensuite comparés avec les nombreux codes convolutionnels doublement orthogonaux perforés proposés dans [7].

4.2 Introduction à la perforation des codes convolutionnels

La perforation est une technique simple permettant d'augmenter le taux de codage en éliminant de façon périodique certains symboles du mot de code généré par le codeur. Ce mécanisme offre comme avantage une diminution du facteur d'expansion de la largeur de bande rendant les communications plus efficaces du point de vue de la largeur de bande. En contre partie, la perforation engendre une diminution des performances d'erreur.

L'élimination des symboles codés s'effectue selon un certain patron de perforation qui est caractérisé par une matrice de perforation $\mathbf{P}$. Cette matrice est constituée

d'éléments binaires où les 0 représentent les symboles perforés et les 1 correspondent aux symboles conservés. Le nombre de lignes et de colonnes composant cette matrice varie en fonction du taux de codage initial et du taux de codage obtenu après perforation. Pour le cas de figure envisagé dans ce mémoire, où les codes d'origine sont tous des codes convolutionnels systématiques de taux $R = \frac{1}{2}$ et où le taux de codage obtenu après la perforation est égal à $R = \frac{b}{b+1}$, la matrice de perforation $\mathbf{P}$ considérée comporte alors 2 rangées et b colonnes, tel qu'illustré en (4.1).

$$\mathbf{P} = \left(\underbrace{\begin{array}{cccccccc} 1 & 1 & \dots & 1 & \dots & 1 & 1 \\ 0 & 0 & \dots & 1 & \dots & 0 & 0 \end{array}}_{b \text{ colonnes}} \right) \tag{4.1}$$

La première ligne ne comporte que des 1 et est associée aux symboles d'information transmis par le codeur systématique, tandis que la deuxième ligne ne contient qu'un seul 1 et indique le seul symbole de parité conservé pour les b symboles d'information transmis vers le canal tous les $(b-1)$ autres symboles de parités sont éliminés. Par convention, la position de la valeur 1 sur la deuxième ligne de la matrice $\mathbf{P}$ est notée π, $0 \leq \pi \leq b-1$, ainsi les symboles de parité conservés sont représentés par l'ensemble suivant [7]:

$$\{p_{\pi+qb}\ ,\ q = 0, 1, 2, \ldots\} \tag{4.2}$$

Exemple 4.1- Considérons un code convolutionnel systématique de taux de codage $R = \frac{1}{2}$ pour encoder la séquence d'information $\mathbf{u}$. On observera à la sortie du codeur la séquence suivante :

$$\begin{aligned} \mathbf{v} &= (v_0, v_1, v_2, v_3, v_4, v_5, v_6, v_7, \ldots) \\ &= ((u_0, p_0), (u_1, p_1), (u_2, p_2), (u_3, p_3), \ldots) \end{aligned}$$

où u_i et p_i représentent les symboles d'information et de parité respectivement. Pour obtenir un taux de codage plus élevé, soit par exemple $R = \frac{2}{3}$, alors on perfore la séquence de sortie avec une matrice $\mathbf{P}$ tel qu'illustrée ci-dessous ($\pi = 0$).

$$\mathbf{P} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \tag{4.3}$$

On transmet alors vers le canal la séquence perforée $\mathbf{v}'$:

$$\begin{aligned} \mathbf{v}' &= ((u_0, p_0), (u_1, X), (u_2, p_2), (u_3, X), \ldots) \\ &= (v_0, v_1, v_2, X, v_4, v_5, v_6, X, \ldots) \end{aligned}$$

où les 'X' représentent les symboles éliminés par le processus de perforation. On observe que pour deux symboles d'information transmis vers le canal, on ne transmet plus 2 symboles de parité mais un seul. Donc le taux de codage obtenu après la perforation est $R = 2/3$ et l'ensemble des symboles de parité conservé est donné par $\{p_{2q} : q = 0, 1, \ldots\}$. Si π avait été égal à 1 alors l'ensemble représentant les symboles de parité conservés aurait été $\{p_{1+2q} : q = 0, 1, \ldots\}$.

4.3 Décodage itératif à seuil adapté aux codes perforés de taux $R = \frac{b}{b+1}$

Intuitivement, nous pouvons envisager que l'action de perforer les codes entraînera une dégradation des performances d'erreur lors du décodage, car le décodeur disposera de moins d'information pour décoder les symboles d'information. Il est bien connu en théorie des communications qu'une augmentation du taux de codage entraîne une augmentation des probabilités d'erreur. En fait ici, l'élimination des symboles de parité conduit à une diminution de l'information extrinsèque, $L_e^{(\mu)}(\hat{u}_i)$,

ce qui entraîne une dégradation des performances d'erreur.

En se référant aux expressions (4.4) et (4.5) on constate que seuls les symboles de parité provenant du canal $y^p_{i+\alpha_k}$ interviennent dans le calcul de l'information extrinsèque pour décoder le symbole $\hat{u}_i$ à l'instant i.

- itération $\mu = 1$

$$\begin{aligned} \lambda_i &= y^u_i + \sum_{k=1}^{J} \psi^{(1)}_{i,k} \\ &= y^u_i + \sum_{k=1}^{J} \left(y^p_{i+\alpha_k} \Diamond \sum_{j=1}^{k-1}{\Diamond} y^u_{i+\alpha_k-\alpha_j} \Diamond \sum_{j=k+1}^{J}{\Diamond} \lambda^{(1)}_{i+\alpha_k-\alpha_j} \right) \end{aligned} \tag{4.4}$$

- itération μ, $\mu \geq 2$

$$\begin{aligned} \lambda^{(\mu)}_i &= y^u_i + \sum_{k=1}^{J} \psi^{(\mu)}_{i,k} \\ \lambda^{(\mu)}_i &= y^u_i + \sum_{k=1}^{J} \left(y^p_{i+\alpha_k} \Diamond \sum_{l=1}^{k-1}{\Diamond} \lambda^{(\mu-1)}_{i+\alpha_k-\alpha_l} \Diamond \sum_{l=k+1}^{J}{\Diamond} \lambda^{(\mu)}_{i+\alpha_k-\alpha_l} \right) \end{aligned} \tag{4.5}$$

Par conséquent, les seuls symboles de parité qui pourront être utilisés pour le calcul des équations $\psi^{(\mu)}_{i,k}$ seront ceux dont l'indice $(i+\alpha_k)$ sera égal à l'indice des symboles de parité conservés après la perforation $(\pi + qb)$, $q = 0, 1, 2, \ldots$ et $k = 1, 2, \ldots, J$. Donc, si la condition suivante est respectée [7]:

$$\alpha_k = (\pi - i) \mathrm{mod}(b), k = 1, \ldots, J \tag{4.6}$$

alors les équations $\psi^{(\mu)}_{i,k}$ pourront bénéficier du symbole de parité provenant du canal $y^p_{i+\alpha_k}$. Autrement, si la position α_k ne répond pas à (4.6), alors les équations $\psi^{(\mu)}_{i,k}$ devront être éliminées lors du décodage pour éviter qu'elles ne modifient de façon aléatoire la valeur de fiabilité associée au symbole décodé. Le décodeur doit donc être modifié pour fixer ces équations $\psi^{(\mu)}_{i,k}$ à zéro.

4.4 Étude des performances d'erreur des codes perforés de taux $R = \frac{b}{b+1}$

Nous venons de mentionner que la perforation entraîne l'élimination de certaines équations de parité. Par conséquent, cette remarque nous amène à définir la distribution des équations de parité $\psi_{i,k}^{(\mu)}$ disponibles dans le temps pour décoder les symboles d'information. Nous définissons le nombre d'équations de parité disponibles pour décoder le symbole u_i à l'instant i à l'itération μ par la cardinalité de l'ensemble suivant [7] :

$$\mathcal{A}_h = \{\alpha_k : \alpha_k = (\pi - i) = h \bmod(b)\} \tag{4.7}$$

où la cardinalité de cet ensemble est définie par $\mathcal{J}_h = |\mathcal{A}_h|$ avec $h = 0, 1, \ldots, b-1$. Nous remarquons pour b et π fixés, l'indice temporel i définit entièrement quel sous-ensemble de connexions, $\mathcal{A}_h$, sera utilisé pour décoder le symbole d'information u_i. D'ailleurs, ce même sous-ensemble sera utilisé pour décoder les symboles d'information u_{i+qb}, $q = 0, 1, \ldots$ et $i = 0, 1, \ldots$. Il est évident que chaque connexion ne fait partie que d'un sous ensemble, $\mathcal{A}_h$, donc, $J = \sum_{h=0}^{b-1} \mathcal{J}_h$. Si la cardinalité de tous les ensembles est égale,

$$|\mathcal{A}_0| = |\mathcal{A}_1| = \ldots = |\mathcal{A}_{b-1}| = \frac{J}{b}$$

alors nous dirons que le code perforé offre une protection égale sur les symboles décodés pour le taux de codage R et sera défini EEPP (Equal Error Punctured Protection). Dans le cas contraire, le code perforé offrira une protection inégale sur les symboles décodés et sera défini UEPP (Unequal Error Punctured Protection). Schématiquement, la distribution des équations s'effectue à l'aide du spectre de perforation des codes. Ce spectre représente la proportion des symboles décodés en fonction du nombre d'équations de parité disponibles. Par exemple, à la figure 4.1a, nous présentons le spectre de perforation d'un code UEPP, et à la figure 4.1b,

celui d'un code EEPP. Nous constatons que les codes EEPP ne comportent qu'une seule raie et que la proportion des symboles décodés avec $\frac{J}{b}$ équations de parité est égale à 1, car tous les symboles sont décodés avec le même nombre d'équations. Le spectre des codes UEPP se différencie du spectre précédent, car il comporte plusieurs raies et la proportion des symboles décodés avec $\mathcal{J}_h$ équations de parité n'est pas nécessairement égale pour tous les symboles.

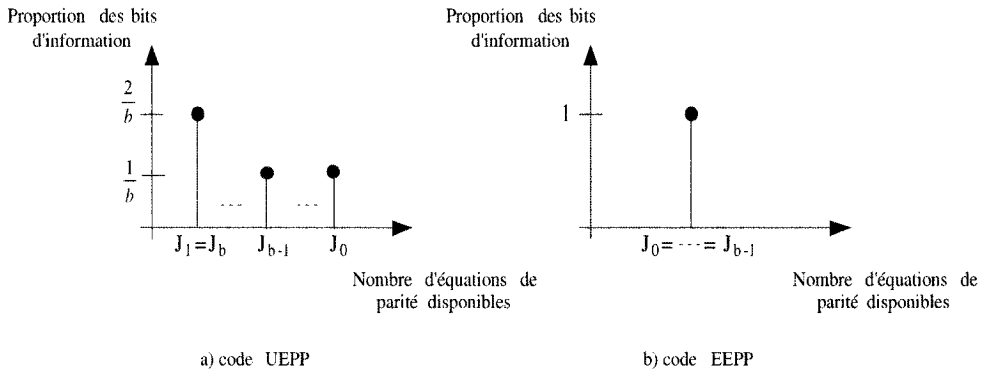

Figure 4.1: Spectres de perforation associés aux codes a) UEPP b) EEPP

Au chapitre précédent, nous avons défini le gain de codage asymptotique du décodeur itératif à seuil à l'aide de la relation (2.29) qui peut être adaptée, pour tenir compte de l'effet de la perforation [7] :

$$G_c = 10\log_{10}(d_{min}R) \tag{4.8}$$

où $d_{min} = J_{min} + 1$. Le nombre d'équations de parité minimum disponibles J_{min} pour décoder un symbole d'information s'obtient avec la relation suivante :

$$J_{min} = \min_{h=0,\ldots,(b-1)}\{\mathcal{J}_h\} \leq \frac{J}{b} \tag{4.9}$$

où J_{min} prend la valeur maximum $\frac{J}{b}$ si le code est de type EEPP. Donc, pour une dimension J fixée, la distance minimum d'un code EEPP sera supérieure à celle d'un code de type UEPP et par conséquent, les performances d'erreur des codes

EEPP seront supérieures à celles des codes UEPP. Pour cette raison, la prospection des codes perforés sera axée uniquement sur la recherche de bons codes EEPP.

4.5 Codes convolutionnels perforés doublement orthogonaux simplifiés au sens large S-PCSO^{2}C-WS

Jusqu'à présent, nous avons énoncé les principaux effets que provoque la perforation des codes sur les performances d'erreur du décodeur itératif à seuil, sans définir les conditions que doivent rencontrer les codes perforés. Dans cette section, nous définissons ces conditions ainsi que les techniques de recherche utilisées pour les obtenir.

La figure 4.2 présente les 2 classes de codes perforés convolutionnels doublement orthogonaux simplifés au sens large (S-PCSO^{2}C-WS) possibles. La figure indique qu'il est possible de former des codes S-PCSO^{2}C-WS offrant une protection identique (EEPP) ou inégale (UEPP) sur les symboles d'information.

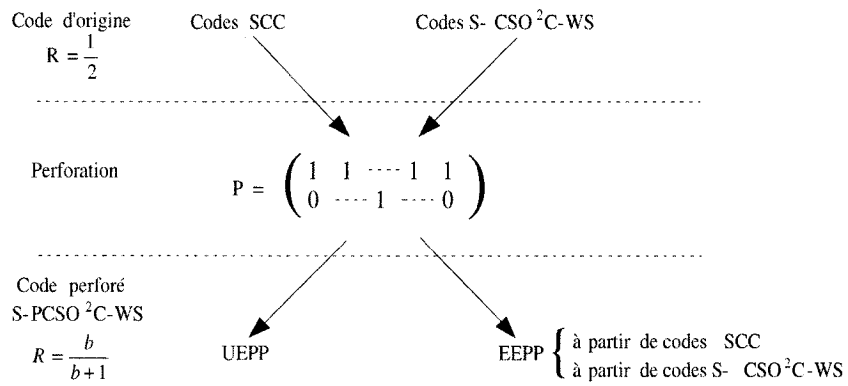

Figure 4.2: Types de codes convolutionnels perforés doublement orthogonaux simplifiés au sens large (S-PCSO^{2}C-WS)

La perforation permet de générer des codes S-PCSO^{2}C-WS de taux de codage $R = \frac{b}{b+1}$ à partir de codes origine de taux de codage initial $R = \frac{1}{2}$ de type S-

CSO^{2}C-WS ou bien à partir de code origine de type convolutionnel systématique (SCC) ne répondant à aucune des conditions de double orthogonalité au départ.

4.5.1 Conditions de double orthogonalité pour les codes perforés

L'équation (3.4) ainsi que la condition (4.6) nous permettent d'établir les conditions sur la double orthogonalité des codes perforés PCSO^{2}C-WS. Celles-ci se résument de la manière suivante [7].

> **Définition 3** : Un code convolutionnel systématique (SCC) de taux de codage $R = \frac{1}{2}$ est doublement orthogonale au sens large pour un taux de codage $R = \frac{b}{b+1}$ (PCSO^{2}C-WS) si et seulement si l'ensemble des connexions du code SCC $\mathcal{A} = \{\alpha_k \ : \ k = 1, 2, \ldots, J \}$ répond aux conditions suivantes :
>
> **1-** L'ensemble des différences simples $S_h = \{(\alpha_k^{(h)} - \alpha_l) \ : \ k \neq l \ \alpha_k \in \mathcal{A}_h, \ \alpha_l \in \mathcal{A}\}$ est composé d'éléments *distincts* ;
>
> **2-** L'ensemble des différences des différences $D_h = \{(\alpha_k^{(h)} - \alpha_l^{(h')}) - (\alpha_m - \alpha_n^{(h')}) : k \neq l, m \neq n, k \neq m, l \neq n, \ h' = 0, 1, \ldots, (b-1), \ \alpha_l, \alpha_n \in \mathcal{A}_{h'}, \alpha_m \in \mathcal{A}\}$ est composé d'éléments *distincts* à l'exception des différences inévitables ;
>
> **3-** L'ensemble des différences simples et l'ensemble des différences des différences doivent être *disjoints*, $S_h \cap D_h = \emptyset$.

Pour générer les codes perforés doublement orthogonaux simplifiés au sens large, nous modifions la deuxième condition de la définition 3 en acceptant dorénavant certaines différences doubles identiques autre que les différences doubles inévitables. Un code S-PCSO^{2}C-WS devra donc répondre aux conditions de la définition 4.

Définition 4 : Un code convolutionnel systématique (SCC) de taux de codage $R = \frac{1}{2}$ est S-PCSO^{2}C-WS si et seulement si l'ensemble des connexions du code SCC $\mathcal{A} = \{\alpha_k \ : \ k = 1, 2, \ldots, J \}$, répond aux conditions suivantes :

1- L'ensemble des différences simples $S_h = \{(\alpha_k^{(h)} - \alpha_l) \ : \ k \neq l \ \alpha_k \in \mathcal{A}_h, \ \alpha_l \in \mathcal{A}\}$ est composé d'éléments *distincts* ;

2- L'ensemble des différences des différences $D_h = \{(\alpha_k^{(h)} - \alpha_l^{(h')}) - (\alpha_m - \alpha_n^{(h')}) \ : \ k \neq l, m \neq n, k \neq m, l \neq n, \ h' = 0, 1, \ldots, (b-1), \ \alpha_l, \alpha_n \in \mathcal{A}_{h'}, \alpha_m \in \mathcal{A}\}$ est composé de $N_d^{e(h)}$ éléments identiques à l'exception des différences des différences inévitables ;

3- L'ensemble des différences simples et l'ensemble des différences des différences doivent être *disjoints*, $S_h \cap D_h = \emptyset$.

- **Facteur de simplification associé aux codes S-PCSO^{2}C-WS**

Tout comme pour les codes S-CSO^{2}C-WS, le facteur de simplification associé aux codes perforés s'exprime, pour un sous-ensemble $\mathcal{A}_h$ de connexions, par le ratio entre le nombre de différences doubles égales $N_d^{e(h)}$ et le nombre total de différences doubles $N_d^{(h)}$ (toujours en excluant les différences doubles inévitables).

$$\delta_h = \frac{N_d^{e(h)}}{N_d^{(h)}}, \ \ 0 \leq \delta < 1 \tag{4.10}$$

Le facteur de simplification δ_h est donc associé à l'ensemble des équations de parité formé par l'ensemble $\mathcal{A}_h$. Tout comme cet ensemble, δ_h est périodique et par conséquent, pour chaque ensemble $\mathcal{A}_h$, $h = 0, 1, \ldots, b-1$, nous aurons un facteur de simplification. Donc, un code S-PCSO^{2}C-WS sera représenté par b facteurs de simplification et nous caractériserons ce code à l'aide du facteur maximum, δ_{max}.

$$\delta_{max} = \max_{h=0,\ldots,b-1} \{\delta_h\} \tag{4.11}$$

- **Spectres de perforation associés aux codes S-PCSO^{2}C-WS**

Nous pouvons modifier le spectre de perforation présenté à la section 4.4 pour tenir compte de la distribution dans le temps des différents taux δ_h entre les équations. La figure 4.3 présente les spectres de perforation associés aux codes perforés de type EEPP et UEPP. Nous remarquons que les codes EEPP n'offrent plus vraiment

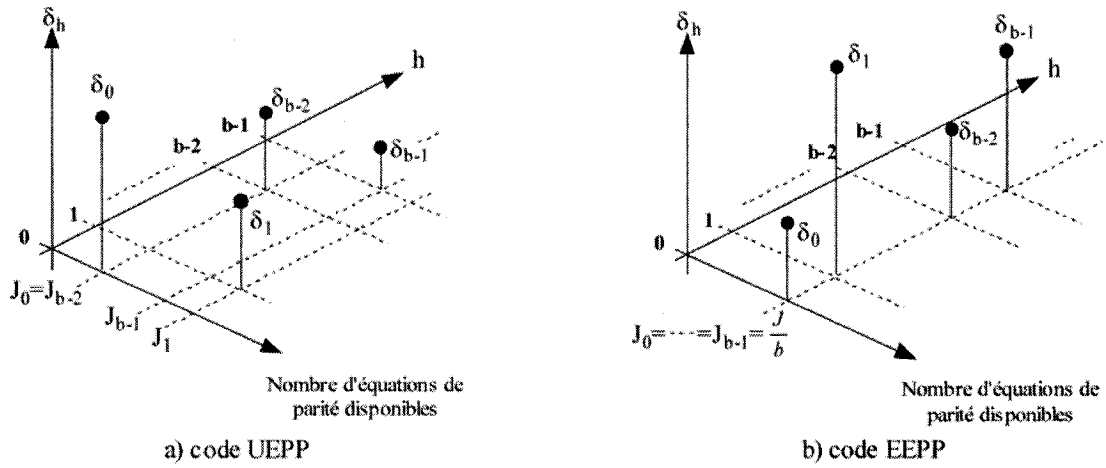

Figure 4.3: Spectre de perforation associé aux codes a) UEPP et b) EEPP

une protection identique sur les symboles décodés, car certains d'entre eux seront décodés avec un ensemble d'équations plus corrélées que d'autres. En observant les raies du spectre des codes EEPP sur une période de b symboles, nous pouvons entièrement déterminer les performances d'erreur du décodeur. Chaque bit étant décodé avec le même nombre d'équations de parité, mais avec un facteur δ_h différent, nous pouvons écrire la probabilité d'erreur moyenne sur les bits décodés comme suit :

$$\overline{P_b} = \frac{1}{b} \sum_{h=0}^{b-1} P_{\delta_h} \tag{4.12}$$

où P_{δ_h} représente la probabilité d'erreur associée au symbole d'information décodé avec J/b équations de parité possédant un facteur de simplification égal à δ_h. Il est clair que la probabilité d'erreur en (4.12) est inférieure à la probabilité d'erreur associée au pire cas possible, soit celui où les symboles sont tous décodés avec le

système d'équations le plus corrélé ($\delta_h = \delta_{max}$).

$$\overline{P_b} < P_{\delta_{max}} \tag{4.13}$$

Toutefois, nous avons remarqué au chapitre précédent qu'à haut SNR les performances d'erreur des codes doublement orthogonaux simplifiés convergeaient vers les performances d'erreur des codes CSO^{2}C-WS et ce quelque soit la valeur du facteur de simplification du code utilisé. À partir de ces résultats on en déduit que la probabilité d'erreur $P_{\delta_{max}}$ converge (à haut SNR) vers la probabilité d'erreur associée aux codes PCSO^{2}C-WS. Il s'en suit que la probabilité d'erreur moyenne $\overline{P_b}$ doit aussi converger vers les performances obtenues avec les codes PCSO^{2}C-WS. Par conséquent, pour un code simplifié de type EEPP, on peut considérer que chaque bit sera décodé avec la même probabilité d'erreur ce qui n'est pas le cas pour les codes de type UEPP.

Exemple 4.2 - Cet exemple permet de mettre en contexte les notions développées jusqu'à présent sur les codes perforés. Supposons que l'on désire connaître le nombre d'équations utilisées par le décodeur pour décoder les symboles $\hat{u}_i$ aux instants $i = 0$ et $i = 1$ lorsque le code perforé S-PCSO^{2}C-WS de taux de codage R $= \frac{2}{3}$ ayant $\mathcal{A} = \{0, 3, 43, 60, 84, 91, 125, 152, 170, 181\}$, est utilisé avec la matrice $\mathbf{P}$ suivante ($\pi = 0$).

$$\mathbf{P} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \tag{4.14}$$

Selon (4.7), le nombre d'équations disponibles pour décoder les symboles s'obtient en générant les ensembles $\mathcal{A}_h$.

$$\mathcal{A}_h = \{\alpha_k = (\pi - i) = h \bmod(b)\}$$

Donc, pour décoder les symboles aux instants $i = 0$ et $i = 1$ nous devons vérifier les connexions appartenant aux ensembles suivants :

$$\mathcal{A}_0 = \{\alpha_k = 0 \bmod(2)\} \quad \text{et} \quad \mathcal{A}_1 = \{\alpha_k = 1 \bmod(2)\}$$

En calculant le modulo-2 sur chacune des connexions, nous obtenons :

$$\begin{array}{ll} \alpha_1 = 0 = 0 \bmod(2) & \alpha_6 = 91 = 1 \bmod(2) \\ \alpha_2 = 3 = 1 \bmod(2) & \alpha_7 = 125 = 1 \bmod(2) \\ \alpha_3 = 43 = 1 \bmod(2) & \alpha_8 = 152 = 0 \bmod(2) \\ \alpha_4 = 60 = 0 \bmod(2) & \alpha_9 = 170 = 0 \bmod(2) \\ \alpha_5 = 84 = 0 \bmod(2) & \alpha_{10} = 181 = 1 \bmod(2) \end{array}$$

de sorte que :

$$\mathcal{A}_0 = \{\alpha_1, \alpha_4, \alpha_5, \alpha_8, \alpha_9\} \quad \text{et} \quad \mathcal{A}_1 = \{\alpha_2, \alpha_3, \alpha_6, \alpha_7, \alpha_{10}\}$$

Avec ce code, le décodeur dispose, à chaque itération, de 5 équations de parité pour estimer les symboles $\hat{u}_0$ et $\hat{u}_1$ offrant ainsi une protection égale sur les symboles décodés. Les équations utilisées pour décoder les symboles $\hat{u}_0$ et $\hat{u}_1$ sont représentées par les ensembles Ψ_0 et Ψ_1 qui possédent un facteur de simplification égal à $\delta_0 = 0.22$ et $\delta_1 = 0.19$ respectivement.

$$\Psi_0 = \{\psi_{0,1}^{(\mu)}, \psi_{0,4}^{(\mu)}, \psi_{0,5}^{(\mu)}, \psi_{0,8}^{(\mu)}, \psi_{0,9}^{(\mu)}\} \tag{4.15}$$

$$\Psi_1 = \{\psi_{1,2}^{(\mu)}, \psi_{1,3}^{(\mu)}, \psi_{1,6}^{(\mu)}, \psi_{1,7}^{(\mu)}, \psi_{1,10}^{(\mu)}\} \tag{4.16}$$

Le spectre de perforation de ce code peut être représenté par celui de la figure 4.4a. Si la position π était égale à 1, alors le symbole $\hat{u}_0$ serait décodé avec les équations de l'ensemble Ψ_1 tandis que le symbole $\hat{u}_1$ serait décodé avec les équations de l'ensemble Ψ_0. Le spectre de perforation de ce code serait alors celui de la figure 4.4b.

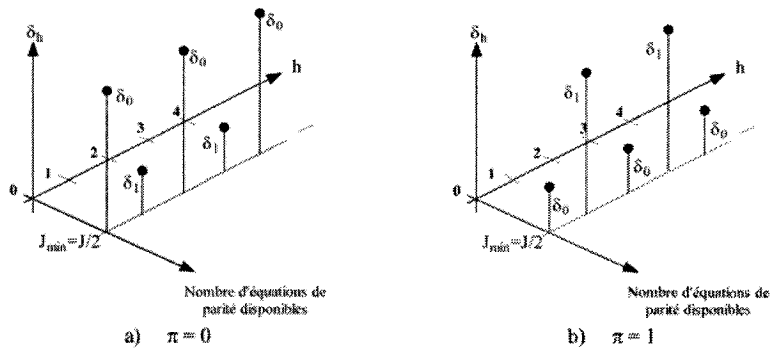

Figure 4.4: Spectres de perforation associés au code de l'exemple 4.2 : a) $\pi = 0$ et b) $\pi = 1$

4.5.2 Recherche des codes perforés simplifiés S-PCSO^{2}C-WS

4.5.2.1 Algorithmes de recherche

Pour effectuer la recherche des codes simplifiés perforés S-PCSO^{2}C-WS, nous avons simplement adapté l'algorithme de recherche aléatoire présenté à la section 3.5.1 du chapitre 3. De cette façon, la structure de l'algorithme présentée à la figure 3.4 est toujours valide à quelques modifications près.

- **Recherche aléatoire des codes perforés S-PCSO^{2}C-WS générés à partir de codes d'origine S-CSO^{2}C-WS**

Cet algorithme de recherche permet la construction de codes S-PCSO^{2}C-WS de type EEPP à partir de codes d'origine S-CSO^{2}C-WS. En remarquant que l'ensemble des conditions imposées sur les codes perforés S-PCSO^{2}C-WS forment un sous-ensemble des conditions imposées aux codes S-CSO^{2}C-WS, nous pouvons donc perforer n'importe quel code de type S-CSO^{2}C-WS et il générera un code S-PCSO^{2}C-WS. Toutefois, rien ne garantit une protection identique sur les symboles décodés. Comme nous l'avons mentionné à la section 4.3, un code perforé offrira une protection EEPP seulement si la cardinalité des ensembles $\mathcal{A}_h$, $h = 0, 1, \ldots, b-1$ est égale à $\frac{J}{b}$. Par conséquent, l'algorithme de recherche aléatoire doit être modifié pour tenir compte de ce critère. Dorénavant, une nouvelle connexion ne sera créée que si elle répond à la condition suivante :

$$\alpha_k = h \bmod(b),\ h = 0, 1, \ldots, b-1$$

et

$$\mid \mathcal{A}_h \mid < \frac{J}{b} \tag{4.17}$$

autrement, si ces conditions ne sont pas respectées, nous retirons la valeur testée de la liste L_k et nous tirons une nouvelle différence simple de façon aléatoire dans la liste jusqu'à ce qu'une nouvelle connexion α_k soit trouvée. Cette modification

représente le seul changement apporté à l'algorithme de départ. Les codes perforés S-PCSO^{2}C-WS de type EEPP obtenus à partir de codes S-CSO^{2}C-WS sont disponibles à la section 4.5.2.2.

- **Recherche aléatoire des codes perforés S-PCSO^{2}C-WS générés à partir de codes d'origine SCC**

Pour obtenir des codes simplifiés S-PCSO^{2}C-WS à partir de codes convolutionnels systématiques SCC, l'algorithme de recherche aléatoire présenté ci-dessus doit être modifié pour respecter la définition 4 énoncée à la section 4.5.1. À la section suivante, nous listons les meilleurs codes simplifiés pour différents taux de codage compris entre $\frac{2}{3}$ et $\frac{6}{7}$.

4.5.2.2 Tableaux des meilleurs codes perforés S-PCSO^{2}C-WS de type EEPP

- **Tableaux des codes perforés S-PCSO^{2}C-WS générés à partir de codes d'origine S-CSO^{2}C-WS**

Les deux tableaux de cette section présentent des codes S-PCSO^{2}C-WS offrant une protection égale sur les symboles décodés. Le tableau 4.2 est particulièrement intéressant, car les codes simplifiés sont de taux compatibles $\frac{1}{2}$, $\frac{2}{3}$ et $\frac{3}{4}$.

Tableau 4.1: Meilleurs ensembles de connexions $\alpha_k \in \mathcal{A}$ pour les codes S-PCSO^{2}C de taux compatibles R=1/2, R=2/3, $\pi = 0$, $10 \leq J \leq 20$

J	δ_{max}	Ensemble des connexions $\{\alpha_k\}$
10	0.080	{0, 6, 66, 75, 143, 307, 317, 356, 397, 420}
12	0.104	{0, 16, 226, 423, 656, 659, 1055, 1110, 1161, 1227, 1239, 1326}
14	0.074	{0, 163, 496, 583, 613, 1074, 1086, 1620, 1917, 2109, 2129, 2278, 2391, 2468}
16	0.079	{0, 221, 491, 758, 828, 1033, 1109, 1276, 1711, 2310, 3115, 3877, 4108, 4230, 4361, 4532}
18	0.081	{0, 75, 280, 376, 853, 1049, 1464, 1957, 2049, 2437, 2754, 3494, 4442, 5152, 5990, 6257, 7419, 7433}
20	0.140	{0, 219, 416, 857, 1047, 1493, 1958, 2419, 2490, 2881, 3385, 4169, 4677, 5364, 6167, 6202, 8310, 10668, 11288, 11762}

Tableau 4.2: Meilleurs ensembles de connexions $\alpha_k \in \mathcal{A}$ pour les codes S-PCSO^{2}C-WS de taux compatibles R=1/2, R=2/3 et R=3/4, $\pi = 0$, $J = 12, 16, 18$

J	R = 2/3	R = 3/4	Ensemble des connexions $\{\alpha_k\}$
	δ_{max}		
12	0.254	0.036	{0, 104, 314, 556, 590, 683, 916, 1059, 1213, 1215, 1233, 1327}
16	0.076	0.040	{0, 84, 433, 531, 804, 928, 939, 1422, 1457, 1787, 1981, 2461 ,3942, 4239, 4522, 4626}
18	0.081	0.070	{0, 75, 280, 376, 853, 1049, 1464, 1957, 2049, 2437, 2754, 3494, 4442, 5152, 5990, 6257, 7419, 7433}

- **Tableaux des codes perforés S-PCSO^{2}C-WS générés à partir de codes d'origine SCC**

Tableau 4.3: Ensembles des meilleures connexions $\alpha_k \in \mathcal{A}$ pour les codes S-PCSO^{2}C-WS, R=2/3, générés à partir de codes SCC, $\pi = 0$, $6 \leq J \leq 18$

J	δ_{max}	Ensemble des connexions $\{\alpha_k\}$
6	0.167	{0, 3, 10, 11, 17, 22}
8	0.157	{0, 5, 12, 31, 51, 66, 67, 68}
10	0.147	{0, 30, 51, 97, 117, 160, 214, 221, 225, 232}
	0.225	{0, 3, 43, 60, 84, 91, 125, 152, 170, 181}
12	0.079	{0, 18, 93, 106, 293, 497, 827, 922, 948, 954, 967, 977}
	0.136	{0, 42, 165, 301, 320, 329, 487, 538, 578, 625,641, 666}
	0.121	{0, 53, 151, 233, 282, 292, 595, 602, 603, 626, 643, 648}
	0.195	{0, 23, 47, 63, 91, 128, 222, 368, 370, 465, 510, 523}
14	0.182	{0, 38, 183, 340, 443, 546, 696, 969, 1234, 1257, 1293, 1319, 1350, 1359}
	0.250	{0, 78, 318, 322, 341, 676, 677, 923, 1066, 1125, 1211, 1299, 1304, 1351}
16	0.123	{0, 97, 477, 490, 958, 1387, 1418, 1455, 1678, 2148, 2411, 2860, 2984, 3015, 3041, 3187}
	0.134	{0, 329, 508, 648, 1071, 1267, 1339, 1371, 1500, 1528, 1988, 2201, 2611, 2670, 2673, 2754}
	0.159	{0, 303, 384, 629, 1282, 1292, 1299, 1306, 1867, 2022, 2329, 2441, 2454, 2525, 2541, 2624}
18	0.132	{0, 235, 497, 992, 1095, 1316, 1776, 1904, 2351, 2389, 2759, 3191, 3678, 3708, 3961, 4307, 4758, 4906}
	0.149	{0, 273, 377, 421, 832, 1117, 1689, 1744, 2270, 2709, 2765, 3270, 3788, 4487, 4534, 4672, 4795, 4902}

Tableau 4.4: Ensembles des meilleures connexions $\alpha_k \in \mathcal{A}$ pour les codes S-PCSO^{2}C-WS,R = 3/4, générés à partir de codes SCC, $\pi = 0$, $9 \leq J \leq 18$

J	δ_{max}	Ensemble des connexions $\{\alpha_k\}$
9	0.149	{ 0, 9, 11, 16, 22, 47, 55, 68, 69 }
12	0.045	{0, 70, 155, 256, 311, 373, 485, 551, 555, 601, 651, 750}
	0.169	{0, 32, 83, 156, 176, 178, 273, 310, 340, 343, 347, 348}
	0.215	{0, 13, 63, 81, 82, 104, 128, 158, 224, 235, 249, 256}
15	0.114	{0, 74, 80, 268, 638, 715, 767, 791, 894, 1386, 1395, 1459, 1501, 1540, 1560}
	0.079	{0, 37, 230, 246, 593, 595, 601, 602, 826, 1089, 1122, 1130, 1177, 1203, 1208}
18	0.131	{0, 139, 359, 547, 718, 772, 1105, 1562, 1952, 2230, 2262, 2625, 3081, 3140, 3560, 3704, 3732, 4092}
	0.082	{0, 62, 101, 483, 817, 1306, 1492, 1737, 1791, 1823, 2004, 2440, 2514, 2681, 2888, 2926, 2930, 2935}
	0.099	{0, 7, 46, 442, 728, 1383, 1398, 1439, 1455, 1985, 2160, 2392, 2632, 2693, 2774, 2809, 2864, 2901}

Tableau 4.5: Ensembles des meilleures connexions $\alpha_k \in \mathcal{A}$ pour les codes S-PCSO^{2}C-WS, R=4/5, générés à partir de codes SCC, $\pi = 0$, $12 \leq J \leq 20$

J	δ_{max}	Ensemble des connexions $\{\alpha_k\}$
12	0.058	{0, 7, 13, 65, 130, 135, 171, 209, 244, 254, 262, 268}
16	0.049	{0, 171, 334, 365, 367, 640, 715, 734, 895, 1377, 1400, 1510, 1517, 1524, 1530, 1565}
	0.088	{0, 30, 253, 258, 377, 471, 495, 515, 592, 634, 661, 870, 893, 908, 999, 1040}
20	0.055	{0, 60, 509, 986, 994, 1203, 1298, 1702, 1915, 1981, 2084, 2195, 2433, 2463, 2704, 3092, 3265, 3546, 3863, 4017}
	0.052	{0, 27, 146, 515, 801, 1397, 1584, 1591, 1610, 1686, 2080, 2334, 2901, 3144, 3199, 3277, 3309, 3316, 3330, 3363}

Tableau 4.6: Ensembles des meilleures connexions $\alpha_k \in \mathcal{A}$ pour les codes S-PCSO^{2}C-WS, R=5/6, générés à partir de codes SCC, $\pi = 0$, $10 \leq J \leq 20$

J	δ_{max}	Ensemble des connexions $\{\alpha_k\}$
10	0.221	{0, 3, 6, 7, 9, 14, 17, 21, 23, 25}
15	0.043	{0, 63, 205, 206, 359, 400, 408, 457, 874, 883, 901, 917, 919, 921, 922}
	0.081	{0, 11, 57, 94, 174, 178, 179, 193, 206, 295, 312, 320, 323, 347, 391}
20	0.038	{0, 83, 226, 371, 702, 825, 1322, 1439, 1516, 1566, 1918, 2125, 2384, 2395, 2444, 2668, 2744, 2893, 3107, 3172}
	0.093	{0, 33, 68, 140, 632, 921, 941, 1358, 1374, 1418, 1481, 1645, 1857, 2217, 2624, 2676, 2687, 2800, 2829, 2834}

Tableau 4.7: Ensembles des meilleures connexions $\alpha_k \in \mathcal{A}$ pour les codes S-PCSO^{2}C-WS, R=6/7, générés à partir de codes SCC, $\pi = 0$, $J = 12, 18$

J	δ_{max}	Ensemble des connexions $\{\alpha_k\}$
12	0.034	{0, 4, 7, 38, 39, 42, 59, 64, 73, 87, 92, 95}
18	0.094	{0, 21, 150, 309, 327, 584, 587, 624, 640, 919, 1004, 1151, 1324, 1330, 1351, 1370, 1381, 1397}

4.5.2.3 Variation de la longueur des codes S-PCSO^{2}C-WS

En comparant la longueur des meilleurs codes S-PCSO^{2}C-WS avec celle des meilleurs codes PCSO^{2}C-WS présentés dans [7] nous constatons que les codes perforés doublement orthogonaux simplifiés offrent une réduction importante de la longueur des codes. Par exemple, les figures 4.5 et 4.6 présentent la variation et la réduction de la longueur des meilleurs codes PCSO^{2}C-WS et S-PCSO^{2}C-WS en fonction de J lorsque $R = \frac{2}{3}$ [1]. Pour cet exemple, nous observons qu'il est possible, lorsque nous fixons la longueur de contrainte, d'augmenter la dimension des codes simplifiés d'au moins 2 ordres de grandeur par rapport aux codes PCSO^{2}C-WS. Cette propriété s'avère avantageuse, car les performances d'erreur varient selon la dimension des codes.

[1] Pour alléger le texte, nous présentons à l'annexe IV la comparaison entre la longueur des codes S-PCSO^{2}C-WS et PCSO^{2}C-WS pour les taux de codage supérieurs à $\frac{2}{3}$.

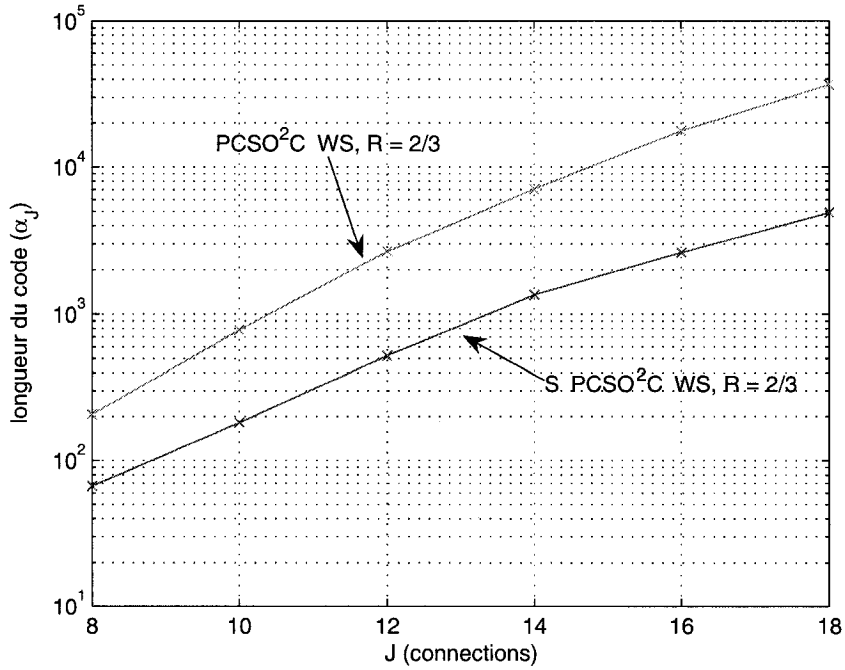

Figure 4.5: Comparaison entre les longueurs les plus courtes pour les codes perforés S-PCSO^{2}C-WS et PCSO^{2}C-WS, $R = \frac{2}{3}$

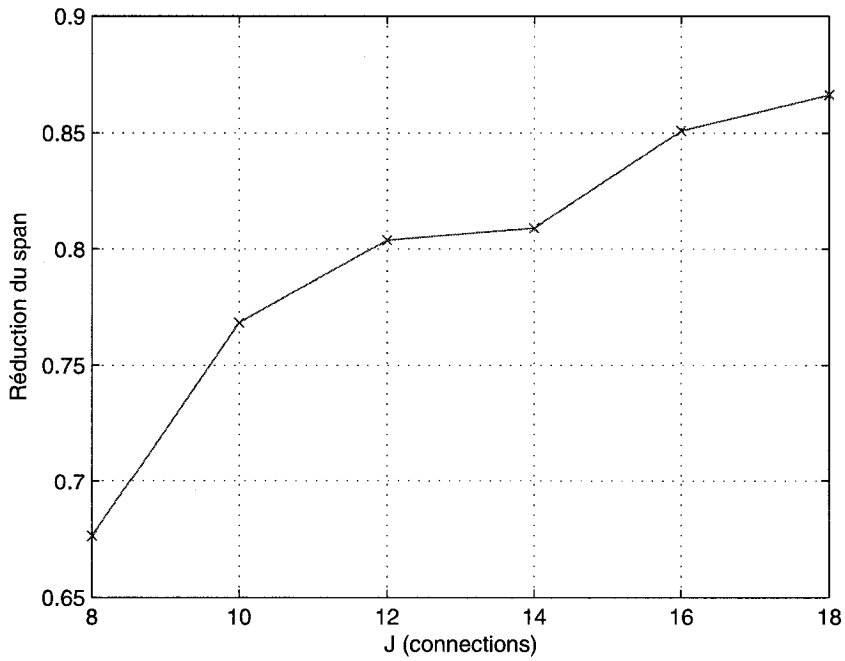

Figure 4.6: Réduction de la longueur entre les codes S-PCSO^{2}C-WS et PCSO^{2}C-WS, $R = \frac{2}{3}$

4.6 Analyse des performances d'erreur des codes S-PCSO^{2}C-WS

4.6.1 Choix des coefficients de pondération

Tout comme pour les codes S-CSO^{2}C-WS l'utilisation de coefficients de pondération lors du décodage des codes perforés et simplifiés permet l'amélioration des performances d'erreur. Les meilleurs coefficients de pondération[2], obtenus pour les codes S-PCSO^{2}C-WS de taux de codage $R = \frac{2}{3}$, sont présentés au tableau suivant.

Tableau 4.8: Meilleurs coefficients a^* pour les codes perforés les plus simplifiés de type EEPP S-PCSO^{2}C-WS, $R = \frac{2}{3}$, $5 \leq J_{\min} \leq 9$, $E_b/N_0 = 4$ dB

	$J_{\min}$				
	5	6	7	8	9
a^*	0.35	0.30	0.30	0.25	0.25

Si nous comparons les coefficients du tableau 4.8 avec ceux du tableau 3.1 nous pouvons observer une faible variation des meilleurs coefficients de pondération. Ces résultats laissent donc croire que le coefficient de pondération semble varier selon le nombre d'équations de parité disponibles pour décoder les symboles d'information et non pas en fonction du type de code doublement orthogonal utilisé. Toutefois, une étude plus approfondie de ces coefficients pourrait nous éclairer à ce sujet. Dans le cadre de ce mémoire, nous nous limiterons à cette discussion.

[2] À la section II.2 de l'annexe II nous présentons les résultats de simulation.

- **Influence de la matrice de perforation sur les performances d'erreur**

Dans cette section, nous analysons l'influence de la position π sur les performances des codes convolutionnels systématiques perforés. La figure 4.7 nous permet de constater que les performances d'erreur obtenues avec les codes S-PCSO^{2}C-WS ne varient pas en fonction de la matrice de perforation.

$$\mathbf{P}_{\pi=0} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \quad \mathbf{P}_{\pi=1} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \tag{4.18}$$

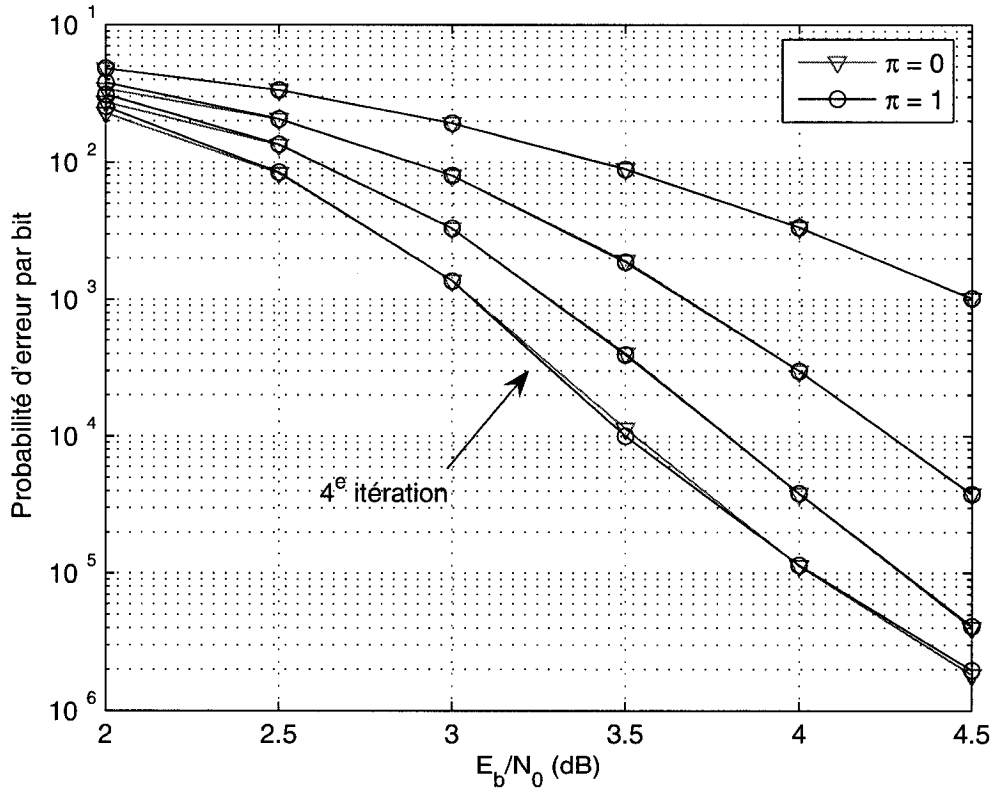

Figure 4.7: S-PCSO^{2}C-WS, $J = 10$, $\mathcal{A} = \{0, 3, 43, 60, 84, 91, 125, 152, 170, 181\}$, $\delta_{max} = 0.225$, $R = \frac{2}{3}$, $a^* = 0.35$

Les résultats obtenus ne sont pas surprenants, car la position π ne fait que permuter les raies du spectre de perforation conservant ainsi, sur une période de b symboles,

la proportion des symboles décodés avec un facteur de simplification δ_h. La figure 4.8 présente les spectres de perforation associés au code de la figure 4.7 lorsque a) $\pi = 0$ ou b) $\pi = 1$. Nous remarquons que tous les symboles sont décodés avec 5 équations de parité. Cependant, lorsque $\pi = 0$ les symboles aux instants i pairs sont moins bien décodés que les symboles aux instant i impairs, car les équations utilisées pour décoder ces symboles comportent plus d'éléments corrélés. À l'inverse, si $\pi = 1$, alors les symboles aux instants i pairs seront mieux décodés que les symboles aux intants i impairs. Donc, quelque soit le scénario envisagé, la probabilité d'erreur moyenne restera la même. Comme les performances d'erreur ne varient pas en fonction de π, toutes les simulations ont été effectuées avec la matrice de perforation suivante, correspondant à $\pi = 0$.

$$\mathbf{P}_{\pi=0} = \left(\underbrace{\begin{matrix} 1 & 1 & \dots & 1 & \dots & 1 & 1 \\ 1 & 0 & \dots & 0 & \dots & 0 & 0 \end{matrix}}_{b \text{ colonnes}} \right) \tag{4.19}$$

où b dépend du taux de codage désiré après perforation $R = \frac{b}{b+1}$.

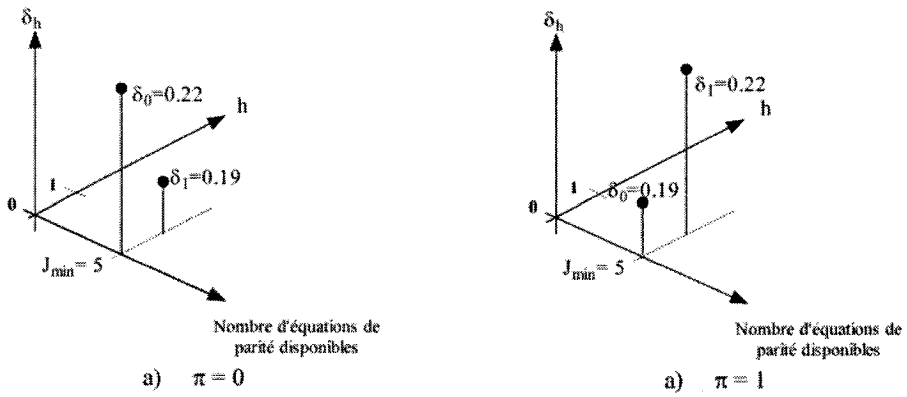

Figure 4.8: Spectres de perforation associés au code S-PCSO^{2}C-WS, $\mathcal{A}$ ={0, 3, 43, 60, 84, 91, 125, 152, 170, 181}, $R = \frac{2}{3}$

- **Influence de δ_{max} sur les performances d'erreur**

La figure 4.9 permet de constater que les performances d'erreur des codes S-PCSO^{2}C-WS convergent vers celles obtenues avec les codes PCSO^{2}C-WS. Ainsi à faible SNR, les codes possédant un facteur δ_{max} plus élevé se comportent moins bien. Toutefois, à plus haut SNR, le comportement des codes S-PCSO^{2}C-WS ne semble plus varier en fonction de δ_{max}. Une observation attentive de ces courbes

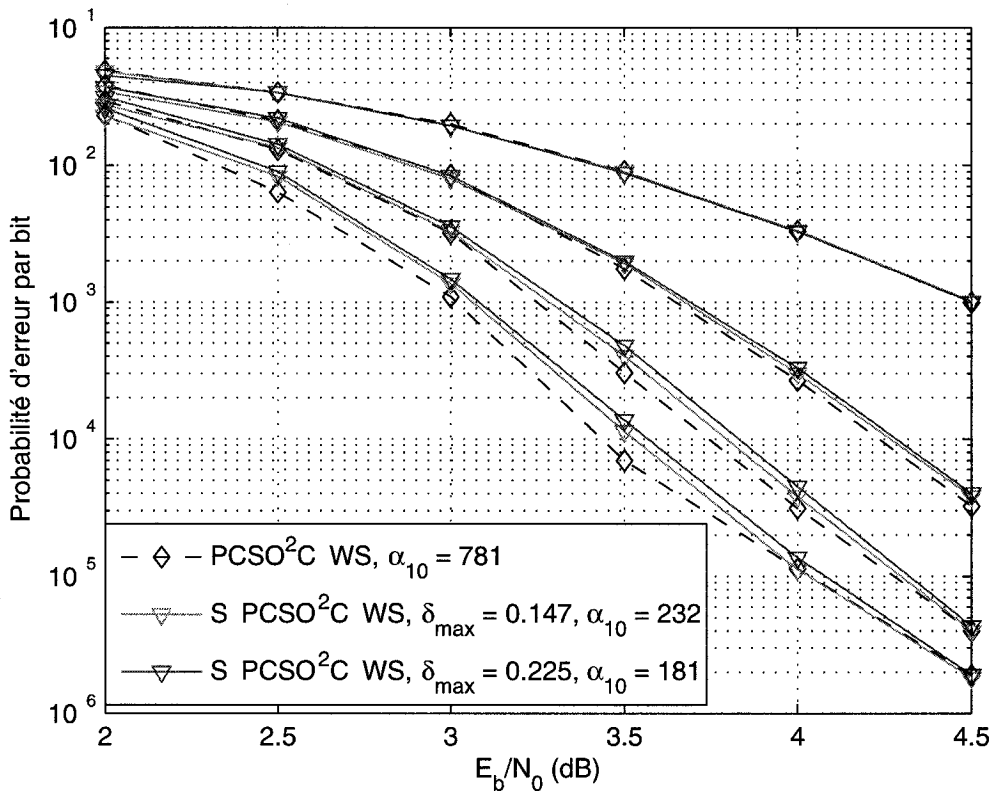

Figure 4.9: Variation des performances d'erreur des codes S-PCSO^{2}C-WS en fonction de δ_{max} pour les itérations 1 à 4, $J = 10$, $R = \frac{2}{3}$, $\pi = 0$

montre que la dégradation des performances d'erreur des codes S-PCSO^{2}C-WS a tendance à s'accentuer au fur et mesure que nous augmentons le nombre d'itérations lorsque le rapport signal sur bruit est faible. L'introduction d'éléments corrélés à la seconde itération explique cette tendance, car l'indépendance entre les observables

est rompue provoquant ainsi une propagation des erreurs. Cette propagation des erreurs est amplifiée d'une itération à l'autre, diminuant ainsi les performances du décodeur comme pour les codes S-CSO^{2}C-WS.

- **Influence du taux de codage sur les performances**

Nous pouvons vérifier à la figure 4.10 la dégradation des performances qu'entraîne l'augmentation du taux codage lorsque la dimension des codes J est fixée. Les codes testés sont ceux du tableau 4.9.

Tableau 4.9: Codes S-CSO^{2}C-WS et S-PCSO^{2}C-WS de type EEPP de dimension $J = 12$

R	d_{min}	δ_{max}	Ensemble des connexions $\{\alpha_k\}$
1/2	13	0.463	{0 1 5 12 32 61 107 271 411 584 707 894}
2/3	7	0.195	{0, 23, 47, 63, 91, 128, 222, 368, 370, 465, 510, 523}
3/4	5	0.169	{0, 32, 83, 156, 176, 178, 273, 310, 340, 343, 347, 348}
4/5	4	0.058	{0, 7, 13, 65, 130, 135, 171, 209, 244, 254, 262, 268}
6/7	3	0.034	{0, 4, 7, 38, 39, 42, 59, 64, 73, 87, 92, 95}

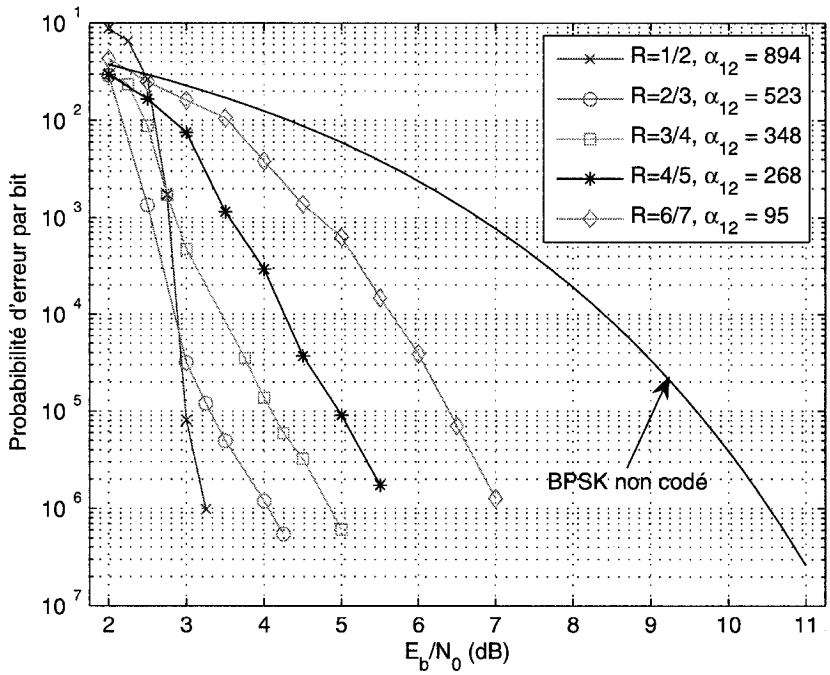

Figure 4.10: Variation des performances d'erreur des codes S-PCSO^{2}C-WS de type EEPP, $J = 12$, 8^e itération, $\frac{1}{2} \leq R \leq \frac{6}{7}$, $\pi = 0$

Nous pouvons justifier ces résultats en analysant l'expression du gain de codage G_c donnée par l'expression (4.8). Nous constatons que pour une valeur de J fixe, une augmentation du taux de codage R entraîne une diminution du nombre d'équations de parité disponibles pour décoder les symboles d'information, diminuant ainsi les performances du décodeur. À l'inverse, si nous fixons le taux de codage, les performances d'erreur s'amélioreront plus la valeur de J sera élevée. La figure 4.11 présente les résultats obtenus pour différentes valeurs de J lorsque R=2/3.

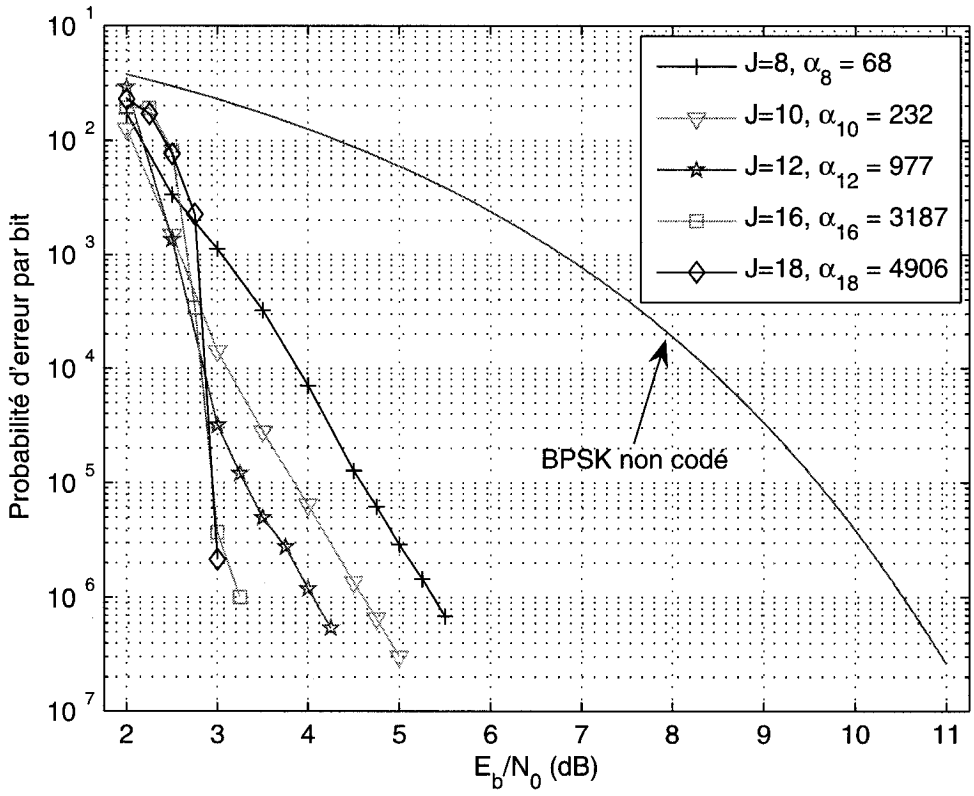

Figure 4.11: Performances d'erreur de codes S-PCSO^{2}C-WS de type EEPP à la 8^e itération, $R = \frac{2}{3}$, $J = 8, 10, 12, 16, 18$

- **Comparaison entre les performances d'erreur des codes perforés S-PCSO^{2}C-WS et PCSO^{2}C-WS**

La figure 4.5 à la section 4.5.2.3 nous a permis de constater que les codes S-PCSO^{2}C-WS offrent une réduction importante de la longueur de contrainte tel qu'il nous est possible, de remplacer un code PCSO^{2}C-WS par un code simplifié de dimension plus élevée, tout en conservant une latence plus faible lors du décodage. Par exemple, les courbes présentées à la figure 4.12 nous permettent de remarquer que le code simplifié de dimension $J = 12$ offre une réduction du délai de décodage total d'environ 33 % par rapport au code PCSO^{2}C-WS de dimension $J = 10$ pour un même nombre d'itérations au décodeur. De plus, le code simplifié offre un gain de codage supplémentaire d'environ 0.25 dB pour une probabilité d'erreur de 10^{-5} et 0.5 dB pour une probabilité d'erreur de 10^{-6}.

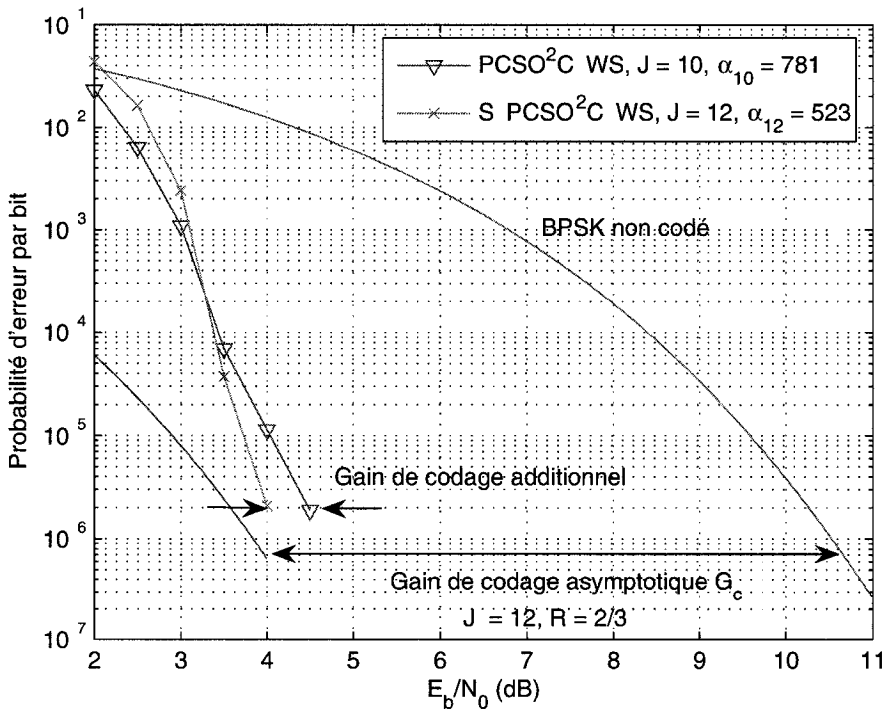

Figure 4.12: Comparaison entre les performances d'erreur des codes S-PCSO^{2}C-WS $J = 12$ et PCSO^{2}C-WS $J = 10$, $R = \frac{2}{3}$, 8^e itération

4.7 Conclusion

Nous avons montré dans ce chapitre l'impact de la perforation des codes sur les performances du décodeur itératif à seuil. Ceci nous a permis de définir les conditions nécessaires que doivent rencontrer les codes convolutionnels perforés doublement orthogonaux simplifiés au sens large. Nous avons aussi vu que ces codes se divisent en deux types offrant : une protection égale ou une protection inégale sur les symboles d'information. Pour chaque type de codes perforés, nous avons développé une représentation spectrale qui nous a permis d'analyser les performances d'erreur. De plus, nous avons présenté deux différentes méthodes pour générer les codes S-PCSO^{2}C-WS soit à partir de codes SCC ou bien à partir des codes S-CSO^{2}C-WS.

Une liste des meilleurs codes perforés a été présentée. Les résultats obtenus nous ont permis de constater que l'utilisation de codes S-PCSO^{2}C-WS permet une réduction importante de la latence totale, lors du décodage par rapport aux codes PCSO^{2}C-WS au détriment d'une légère dégradation des performances à faible rapport signal sur bruit. Toutefois, comme nous l'avons montré, ce léger désavantage peut être contourné en utilisant des codes S-PCSO^{2}C-WS de dimension plus élevée sans pour autant augmenter le délai de décodage.

CHAPITRE 5

CONCLUSION

5.1 Bilan de la recherche réalisée

Ce mémoire, qui s'inscrit dans la lignée des travaux précédemment entamés sur les codes correcteurs d'erreur dits convolutionnels doublement orthogonaux, a permis de mesurer l'influence que provoque une diminution des contraintes imposées aux codes sur les performances d'erreur ainsi que sur la latence lors du décodage.

Suite à nos recherches, nous avons constaté que la réduction des conditions imposées aux codes engendre (en comparaison avec les codes doublement orthogonaux au sens large) :

- une analyse théorique des performances plus difficile dû à la corrélation des équations de parité ;
- une réduction du temps de recherche des bons codes ;
- une réduction importante de la latence lors du décodage ;
- une réduction des composants nécessaires à la réalisation matérielle du décodeur ;
- une faible dégradation des performances d'erreur (à peine quelques dixièmes de décibels).

Il en résulte que les codes simplifiés au sens large S-CSO^{2}C-WS et les codes perforés simplifiés S-PCSO^{2}C-WS répondent de façon satisfaisante à la problématique de latence associée au décodage itératif des codes CSO^{2}C-WS. Ainsi, les codes simplifiés pourraient être utilisés dans des systèmes de communications à haut débit

nécessitant un faible délai de décodage et des performances d'erreur se situant à des valeurs de $\frac{E_b}{N_o}$ entre celles des codes turbo et celles des techniques conventionnelles de codage.

5.2 Améliorations envisageables

Dans un premier temps, nous pourrions étendre la recherche des codes convolutionnels doublement orthogonaux simplifiés et simplifiés perforés au sens large pour des dimensions supérieures à 20 et pour des taux de codage supérieurs à 5/6. La recherche de codes simplifiés perforés de type UEPP n'offrant pas une protection identique sur les symboles d'information décodés pourrait être aussi entreprise.

Deuxièmement, une étude rigoureuse des coefficients de pondération pourrait être entreprise. Cette recherche permettrait l'amélioration des performances d'erreur des codes simplifiés. Nous pourrions aussi étudier l'impact qu'aurait l'élimination des équations de parité comportant un nombre élevé de différences doubles égales.

Par la suite, nous pourrions réduire les conditions d'orthogonalité imposées aux codes définis au sens strict. La recherche de ces codes pourrait être avantageuse, car leurs performances d'erreur sont supérieures à celles des codes définis au sens large.

Nous pourrions aussi envisager l'étude du comportement des performances d'erreur des codes simplifiés, mais cette fois-ci décodés à l'aide d'un algorithme de décodage plus performant tel que l'algorithme de rétropropagation de l'erreur (Belief Propagation).

De plus, nous pourrions envisager l'étude du comportement des performances d'erreur des codes simplifiés en éliminant la boucle de rétroaction (le feedback) à chacune des itérations.

5.3 Ouverture

L'utilisation de codes simplifiés pourrait s'avérer avantageuse pour des canaux à évanouissements. Ce type de canal a tendance à introduire des blocs d'erreur de sorte que les erreurs ne peuvent plus êtres considérées indépendantes les unes des autres. Il serait possible de combattre ce phénomène en entrelaçant les symboles à la sortie du codeur. Toutefois, pour éviter l'utilisation d'un entrelaceur à l'émetteur et au récepteur, on pourrait simplement envisager la multiplication des connexions d'un code par la longueur moyenne des salves d'erreurs. Ceci aurait pour effet d'augmenter la distance entre les symboles orthogonaux constituant les équations de parité. De cette façon, les erreurs pourraient êtres considérées indépendantes les unes des autres. Donc, pour minimiser le délai de décodage les codes possédant une faible mémoire seront avantagés d'où l'utilisation des codes simplifiés.

BIBLIOGRAPHIE

[1] Baechler, B., "Analyse et détermination de codes doublement orthogonaux pour décodage à seuil itératif", *Mémoire de Maîtrise de l'École Polytechnique de Montréal*, Mai, 2000.

[2] Bahl, L., Cocke, J., Jelinek, F. et Raviv, J., "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate", *IEEE Trans. Inform. Theory*, vol. IT-20, pp. 284-287, Mars,1979.

[3] Berrou, C., Glavieux, A. et Thitimajshima P., "Near Shannon Limit Error Correcting Coding and Decoding: Turbo Codes", *Proceedings of ICC'93*, pp. 1064-1070, Mai,1993.

[4] Cardinal, C., "Décodage à seuil itératif sans entrelacement des codes convolutionnels doublement orthogonaux", *Thèse de Doctorat, École Polytechnique de Montréal*, Juin, 2001.

[5] Cardinal, C., Haccoun, D., Gagnon, F. et Batani, N., "Turbo Decoding Using Convolutional Self Doubly Orthogonal Codes", *Proceedings of ICC'99*, pp. 113-117, Juin, 1999.

[6] Provost, G., Cantin, M.A., Sawan, M., Cardinal, C., Savaria, Y., Haccoun, D., "Fast Parameters Optimization of an Iterative Decoder using a Configurable Hardware Accelerator", *IEEE International Symposium on Circuits and Systems*, vol 4, pp. 4159-4162, Mai, 2005.

[7] Haccoun, D., Cardinal, C., "High-Rate Punctured Convolutionnal Self-Doubly Orthogonal Codes for Iterative Threshold Decoding", *IEEE Trans. Commun*, vol 53, pp. 55-63, Janvier, 2005.

[8] Haccoun, D., Cardinal, C., Gagnon, F., "Search and Determination of Convolutional Self-Orthogonal Codes for Iterative Threshold Decoding", *IEEE Trans. Commun*, vol 53, pp. 802-809, Mai, 2005.

[9] Haccoun, D., He, Y.-C., "An Analysis of the Orthogonality Structures of Convolutional Codes for Iterative Decoding", *IEEE Trans. Information Theory*, vol 51, no. 9, pp. 3247-3261, Septembre, 2005.

[10] Gagnon, F., Haccoun, D., Batani, N. et Cardinal, C., "Apparatus for Convolutional Self Doubly Orthogonal Codes for Encoding and Decoding", *US Patent No 6167552*, Décembre, 2002.

[11] Hagenauer, J. et Hoeher, P., "A Viterbi Algorithm with Soft Decision Outputs and its Applications", *Proceedings of Globecom'89*, pp. 1680-1686, Novembre, 1989.

[12] Lin, S. et Costello, D. J. Jr, *Error Control Coding*, Second Edition, Prentice-Hall, Englewood Cliffs, N.J., 2004.

[13] Massey, J. L., *Threshold Decoding*, MIT Press, Cambridge, MA, 1963.

[14] Proakis, J., *Digital Communications*, Third Edition, McGraw-Hill, New York, 1995.

[15] Shannon, C. E., "A Mathematical Theory of Communication", *Bell Syst. Tech. J.*, vol. 27, pp. 379-423, Juillet, 1948.

[16] Shannon, C. E., "A Mathematical Theory of Communication", *Bell Syst. Tech. J.*, vol. 27, pp. 623-656, Juillet, 1948.

[17] Wozencraft, J. M. et Jacobs, I. M., *Principles of Communication Engineering*, Wiley, New York, 1965.

Annexe I

Bornes sur les codes

I.1 Définitions des ensembles

Avant de définir les bornes sur les codes, nous rappelons les différents ensembles associés aux codes doublement orthogonaux.

I.1.1 Ensemble des connexions

L'ensemble des connexions, $\mathcal{A}$, est représenté par J nombres entiers positifs de sorte que $\alpha_1 < \alpha_2 < \ldots < \alpha_J$ où $\alpha_1 = 0$.

I.1.2 Ensemble des différences simples

L'ensemble, $\mathcal{S}$, est représenté par $J(J-1)$ nombres entiers de sorte que $\mathcal{S} = \{s_{i,j} = (\alpha_i - \alpha_j),\ i \neq j\}$. Si $i > j$ alors $s_{i,j} > 0$ et $s_{i,j} \in S_+$, autrement si $j > i$ alors $s_{i,j} < 0$ et $s_{i,j} \in S_-$. On remarque alors que $\mathcal{S} = \mathcal{S}_+ \cup \mathcal{S}_-$ et $N_s = \mid \mathcal{S}_+ \mid = \frac{J(J-1)}{2}$.

I.1.3 Ensemble des différences des différences

L'ensemble des différences des différences est représenté par :

$$D = \{d^{k,l}_{m,n} = (\alpha_k - \alpha_l) - (\alpha_m - \alpha_n) : k \neq l,\ m \neq n,\ m \neq k,\ l \neq n\}$$

Cet ensemble contient toutes les différences doubles possibles, incluant les différences doubles inévitables. Il est montré dans [8] que les conditions imposées aux indices $k,\ l,\ m,\ n$ peuvent êtres réduites à $k \neq l,\ m \neq n,\ m \neq k,\ l \neq n,\ m \leq k,\ n \leq l$ pour les codes CSO^{2}C-WS. Ces nouvelles conditions permettent l'élimination des différences doubles inévitables. Sous ces conditions, nous

définissons l'ensemble équivalent des différences des différences :

$$D^* = \{d_{m,n}^{k,l} = (\alpha_k - \alpha_l) - (\alpha_m - \alpha_n) : k \neq l,\ m \neq n,\ k \neq m,\ l \neq n,\ k \geq n,\ l \geq m\}$$

Le nombre de différences doubles positives dans cet ensemble est donné par [8]:

$$N_d = \mid D_+^* \mid = \frac{1}{8}(J^4 - 2J^3 + 3J^2 - 2J) \tag{I.1}$$

I.2 Borne supérieure sur le facteur de simplification δ

Au chapitre 3 nous mentionnons que la borne supérieure sur le facteur de simplification est donnée par l'expression suivante :

$$\delta \leq 1 - \frac{N_s}{N_d} \tag{I.2}$$

Preuve

Pour prouver l'expression I.2 il suffit d'observer qu'il y a au moins N_s différences doubles positives distinctes. En fait, en constatant qu'il est toujours possible de former les différences doubles suivantes :

$$\begin{aligned} d_{l,k}^{k,l} &= 2(\alpha_k - \alpha_l), \quad k > l \\ &= 2s_{k,l} \end{aligned} \tag{I.3}$$

On remarque que ces différences doubles sont formées à partir de N_s différences simples différentes (car le code doit au moins avoir des différences simples distinctes) donc les différences doubles provenant de (I.3) sont distinctes. Il s'en suit que le nombre maximum de différences doubles égales positives $N_{d,max}^{e}$, en excluant les différences doubles inévitables, est donné par :

$$N_{d,max}^{e} = N_d - N_s$$

alors,

$$\begin{aligned} \delta_{max} &= \frac{N^e_{d,max}}{N_d} \\ &= \frac{N_d - N_s}{N_d} \\ &= 1 - \frac{N_s}{N_d} \\ &< 1 \end{aligned}$$

Mais,

$$N^e_d \leq N^e_{d,max}$$

alors,

$$\delta \leq \delta_{max} < 1$$

■

On peut remarquer au tableau suivant que la borne supérieure n'est pas très serrée par rapport aux facteurs de simplification maximum des codes obtenus.

Tableau I.1: Comparaison entre les facteurs de simplifications maximum des codes S-CSO^{2}C trouvés et la borne supérieure δ_{max}, $5 \leq J \leq 10$

J	5	6	7	8	9	10
Code S-CSO^{2}C max$\{\delta\}$	0.3818	0.4333	0.4416	0.4828	0.4895	0.4917
Borne δ_{max}	0.8182	0.8750	0.9091	0.9310	0.9459	0.9565

On peut expliquer cette différence du fait qu'il y a plus que N_s différences doubles nécessairement distinctes. Une borne plus serrée pourrait nous servir d'indicateur à savoir si les codes simplifiés trouvés à l'aide de l'algorithme de recherche aléatoire (donc les codes de dimension J supérieure à 8) sont près d'une solution optimale. Il reste que la solution à ce problème réside dans un exercice de dénombrement complexe et ne sera pas analysé dans ce mémoire.

I.3 Borne inférieure sur la longueur des codes en fonction du facteur de simplification δ

À la section 3.5.3 du chapitre 3, nous mentionnons que la longueur minimum associée aux codes simplifiés est égale à α_J^* :

$$\alpha_J^* = \lceil \frac{N_s + N_d - N_d^e}{2} \rceil = \lceil \frac{N_s + (1-\delta)N_d}{2} \rceil \tag{I.4}$$

Preuve

Sachant que le nombre de différences simples possibles est donné par N_s et que le nombre de différences doubles, sans les répétitions provenant des permutations d'indices, est N_d et qu'il y a N_d^e différences doubles égales, en excluant les différences inévitables, alors $N_s + N_d - N_d^e$ représente le nombre de différences qui doivent êtres différentes. Or, on remarque que la plus grande différence provient de l'ensemble des différences doubles et est égale à $(\alpha_J - 0) - (0 - \alpha_J) = 2\alpha_J$. Par conséquent, la plus grande différence doit être égale à $N_s + N_d - N_d^e$ d'où I.4■

Note : nous remarquerons que l'expression I.4 représente la borne inférieure associée aux codes CSOC et CSO^{2}C-WS lorsque $\delta = 1$ et $\delta = 0$ respectivement.

I.4 Bornes supérieures sur les connexions d'un code connaissant α^{max}

Dans cette section, nous définissons les bornes utilisées pour effectuer la recherche des codes. Ces bornes nous permettent de limiter notre recherche des bons codes à l'aide des heuristiques développés.

Soit un code, $\mathcal{A}$, représenté par un ensemble de J connexions, $\{\alpha_k\}$, répondant aux critères des codes S-CSO^{2}C-WS et soit α_{max} la valeur maximale que peut prendre la connexion α_J du code $\mathcal{A}$.
On peut représenter la valeur α_J à l'aide de la relation suivante :

$$\begin{aligned} \alpha_J &= (\alpha_J - \alpha_k) + (\alpha_k - \alpha_1) \\ &= s_{J,k} + s_{k,1} \end{aligned}$$

avec $\alpha_1 = 0$, $1 < k < J$ et où $s_{J,k}$ et $s_{k,1}$ représentent des différences simples. Or, comme ce code répond à la définition des codes S-CSO^{2}C-WS toutes les différences simples doivent êtres différentes de sorte que $s_{J,k} \neq s_{k,1}$. Alors, il y a deux possiblités : $s_{J,k} > s_{k,1}$ ou bien $s_{J,k} < s_{k,1}$. Donc si, $s_{J,k} > s_{k,1}$, alors :

$$\alpha_k < \frac{\alpha_J}{2} \tag{I.5}$$

autrement si $s_{J,k} < s_{k,1}$ alors $\alpha_k > \frac{\alpha_J}{2}$. Donc, toutes les connexions $\{\alpha_k\}$, $1 \leq k \leq J$, sont plus grandes ou plus petites que $\frac{\alpha_J}{2}$.
Définissons à présent le code $\mathcal{A}^s$, appelé code symétrique au code $\mathcal{A}$, définit par l'ensemble des connexions suivantes $\mathcal{A}^s = \{\alpha'_l : \alpha'_l = \alpha_J - \alpha_{J+1-l}, l = 1, \ldots, J\}$ (cette transformation est simplement une symétrie miroir du code $\mathcal{A}$ suivie d'une translation de α_J). On remarque que $\alpha'_J = \alpha_J$. Donc, la connexion α'_l est soit plus grande ou bien plus petite que $\frac{\alpha_J}{2}$. Montrons alors que si $\alpha_{\frac{J}{2}}$ est supérieure à $\frac{\alpha_J}{2}$ alors $\alpha'_{\frac{J}{2}}$ est

inférieure à $\frac{\alpha_J}{2}$.

$$\begin{aligned} \alpha'_{\frac{J}{2}} &= \alpha_J - \alpha_{J+1-\frac{J}{2}} \\ &= \alpha_J - \alpha_{\frac{J}{2}+1} \\ &< \alpha_J - \alpha_{\frac{J}{2}} \end{aligned} \tag{I.6}$$

l'inégalité provient du fait que $\alpha_{\frac{J}{2}+1} > \alpha_{\frac{J}{2}}$ car $\alpha_i < \alpha_{i+1}$ (provient de la propriété des ensembles). Donc si $\alpha_{\frac{J}{2}} > \frac{\alpha_J}{2}$ alors I.6 devient :

$$\begin{aligned} \alpha'_{\frac{J}{2}} &< \alpha_J - \alpha_{\frac{J}{2}} \\ &< \frac{\alpha_J}{2} \end{aligned} \tag{I.7}$$

Donc nous venons de montrer que si la position $\alpha_{\frac{J}{2}}$ du code $\mathcal{A}$ est supérieure à $\frac{\alpha_J}{2}$ alors la connexion correspondante dans l'ensemble $\mathcal{A}^s$ sera inférieure à $\frac{\alpha_J}{2}$. Donc, pour tous les codes de types S-CSO^{2}C (en fait l'important c'est que les différences simples soient différentes les unes des autres) on peut limiter notre recherche aux codes ayant $\alpha_{\frac{J}{2}} < \frac{\alpha_J}{2}$. Car si nous trouvons un code pour lequel $\alpha_{\frac{J}{2}} > \frac{\alpha_J}{2}$ alors il y a un code symétrique qui offrira les mêmes propriétés que le code $\mathcal{A}$ et où $\alpha'_{\frac{J}{2}}$ sera inférieure à $\frac{\alpha_J}{2}$.

Maintenant, nous pouvons donc considérer uniquement que le sous-ensemble composé des $J/2$ premiers éléments de $\mathcal{A}$ borné par la valeur $\frac{\alpha_J}{2}$. Par conséquent, en utilisant (I.7) nous avons $\alpha_{\frac{J}{4}} < \frac{\alpha_J}{4}$. En continuant ce raisonnement, nous pouvons établir les bornes à l'aide de la relation suivante :

$$\alpha_{\lceil \frac{J}{2^i} \rceil} < \frac{(\alpha_J^{max} - 1)}{2^i} = B_{\lceil \frac{J}{2^i} \rceil} \tag{I.8}$$

pour $i = 0, \ldots, \lceil log_2(J) \rceil - 1$.

Exemple I.1 - Sachant que le plus petit code CSO^{2}C-WS de dimension $J = 10$ est égal à $\mathcal{A}$={0, 29, 40, 43,1020,1328,1495,1606,1696,1698}. En utilisant la relation (I.8) nous pouvons trouver les bornes utilisées par nos algorithmes de recherche.

Tableau I.2: Valeurs utilisées comme bornes sur les connexions d'un code lorsque $\alpha_{10}^{max} = 1698$ et $J = 10$

i	$\alpha_{\lceil \frac{J}{2^i} \rceil}$	$B_{\lceil \frac{J}{2^i} \rceil} = \frac{(\alpha_J^{max}-1)}{2^i}$
0	α_{10}	1697
1	α_5	849
2	α_3	425
3	α_2	213

Il s'en suit que les bornes sur les connexions de l'ensemble $\mathcal{A}$ sont les suivantes : $\{\alpha_1 = 0, \alpha_2 < \frac{1698}{8}, \alpha_3 < \frac{1698}{4}, \alpha_4 < \alpha_5, \alpha_5 < \frac{1698}{2}, \alpha_6, \alpha_7, \alpha_8, \alpha_9, \alpha_{10} < 1697\}$

Annexe II

Détermination des coefficients de pondération

Cette annexe présente les coefficients de pondération optimaux obtenus lors de simulations pour certains codes simplifiés S-CSO^{2}C-WS et S-PCSO^{2}C-WS. Seuls les résultats associés aux codes les plus simplifiés des tableaux 3.3, 3.4 et 4.3 sont exposés.

II.1 Codes S-CSO^{2}C-WS

La figure II.1 présente la variation des performances d'erreur d'un code S-CSO^{2}C-WS en fonction du coefficient de pondération utilisé sur 8 itérations. Les figures II.2 et II.3 montrent cette même variation des performances d'erreur, mais à la huitième itération uniquement.

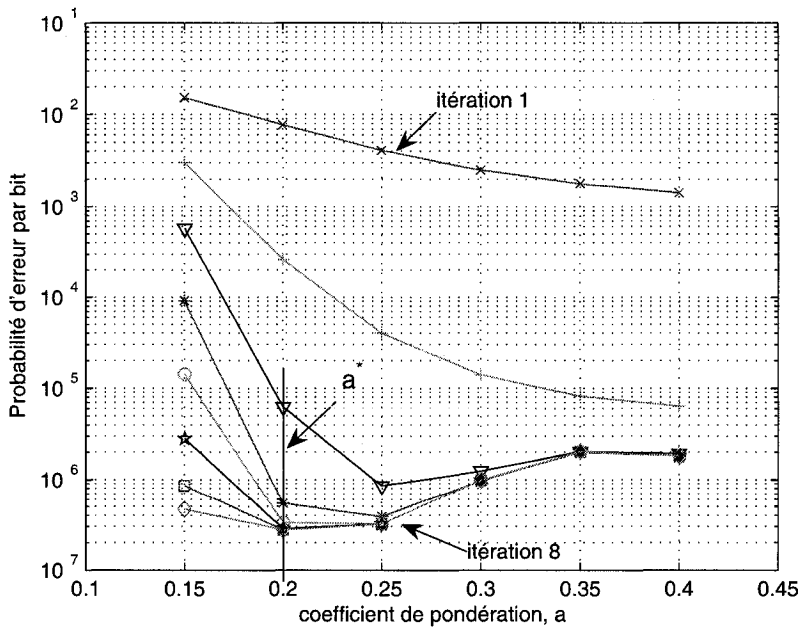

Figure II.1: Probabilité d'erreur en fonction du coefficient de pondération, a, $J = 10$, $R = \frac{1}{2}$, $E_b/N_0 = 4$ dB

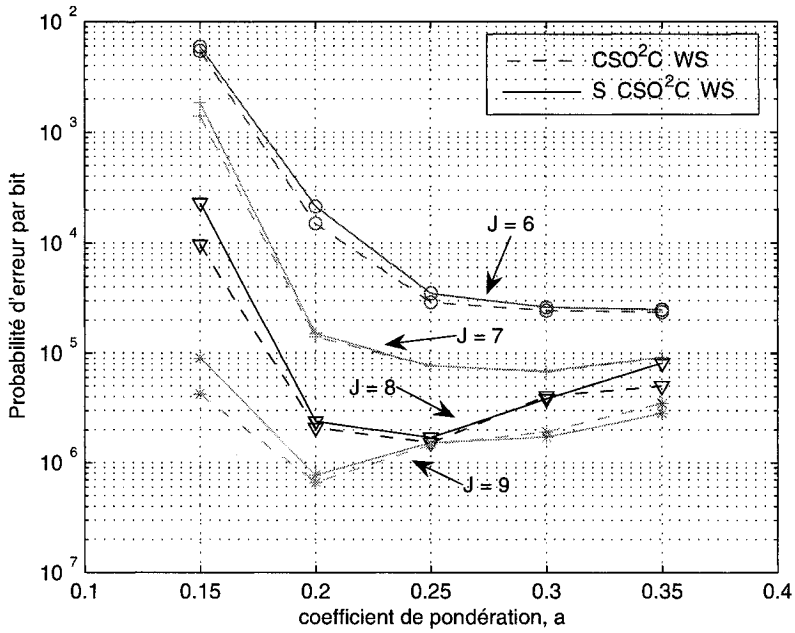

Figure II.2: Probabilité d'erreur en fonction du coefficient de pondération, a, 8^e itération, $5 < J < 10$, $R = \frac{1}{2}$, $E_b/N_o = 4\ dB$

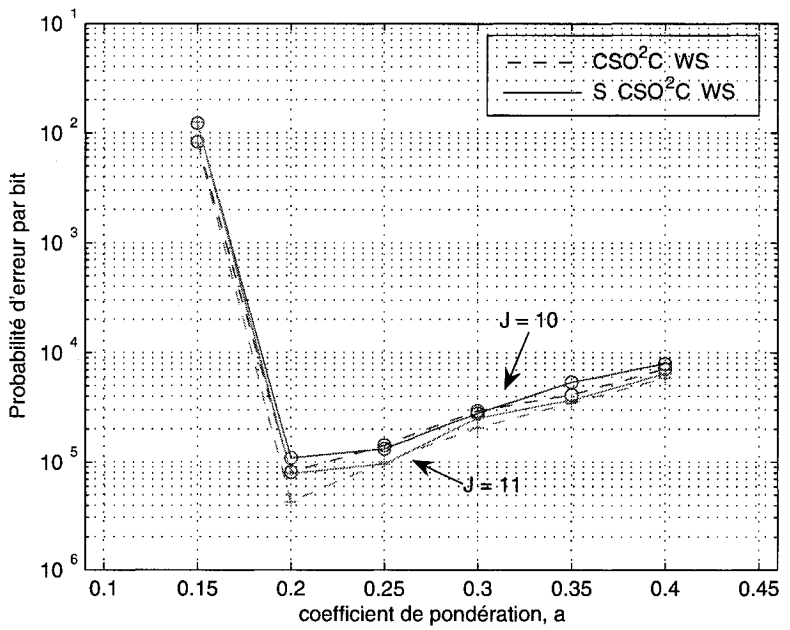

Figure II.3: Probabilité d'erreur en fonction du coefficient de pondération, a, 8^e itération, $9 < J < 12$, $R = \frac{1}{2}$, $E_b/N_o = 4\ dB$

II.2 Codes S-PCSO^{2}C-WS de taux de codage $R = \frac{2}{3}$

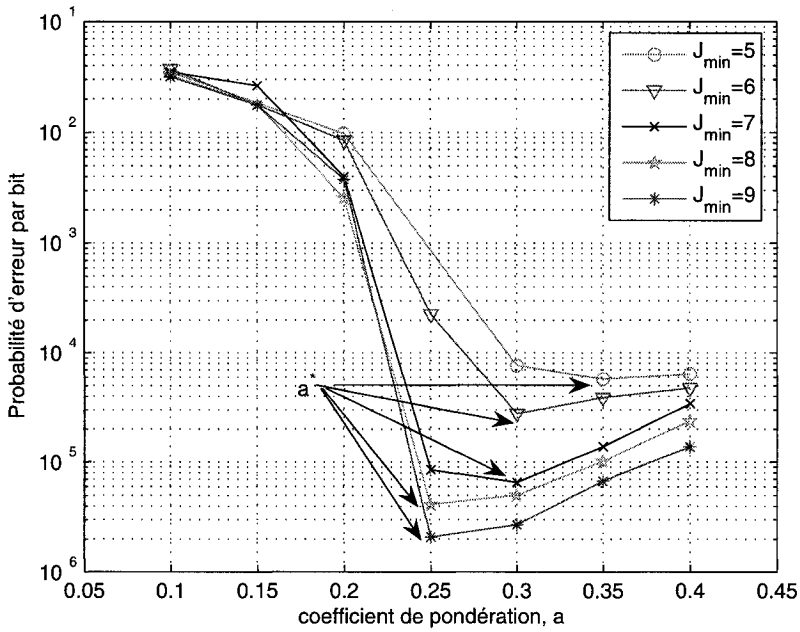

Figure II.4: Probabilité d'erreur en fonction du coefficient de pondération, a, 8^e itération, $4 < J_{min} < 10$, $R = \frac{2}{3}$, $E_b/N_o = 4\ dB$

Annexe III

Répartition des différences double égales selon les J équations de parité

En rappelant que les équations de parité à la seconde itération s'obtiennent avec la relation suivante :

$$\psi_{i,k}^{(2)} = y_{i+\alpha_k}^{p} \Diamond \sum_{l=k+1}^{J} \Diamond \lambda_{i+\alpha_k-\alpha_l}^{(2)} \Diamond \sum_{l=1}^{k-1} \Diamond \Big(y_{i+\alpha_k-\alpha_l}^{u} + \sum_{n=1, n\neq l}^{J} \Big(y_{i+\alpha_k-\alpha_l+\alpha_n}^{p} \Diamond \sum_{m=1, m\neq k}^{n-1} \Diamond y_{i+(\alpha_k-\alpha_l)-(\alpha_m-\alpha_n)}^{u} \Diamond \sum_{m=n+1}^{J} \Diamond \lambda_{i+(\alpha_k-\alpha_l)-(\alpha_m-\alpha_n)}^{(1)} \Big) \Big) \qquad \text{(III.1)}$$

où $k = 1, 2, \ldots, J$. Pour un code S-CSO^{2}C-WS donné, nous pouvons mesurer le nombre de différences doubles positives appartenant à l'ensemble des différences doubles égales, D_+^e, pour chacune des équations de parité. Le tableau III.1 présente cette répartition pour trois codes S-CSO^{2}C-WS de dimension J égale à 10, mais possédant un facteur de simplification δ différent.

Tableau III.1: Répartitions des différences double égales selon les équations de parité

α_{10}	k										$N_d^e = \mid D_+^e \mid$	δ
	1	2	3	4	5	6	7	8	9	10		
340	0	0	0	0	1	16	50	87	142	213	509	0,4917
454	0	0	0	0	2	15	38	76	84	155	370	0,3575
1190	0	0	0	0	0	1	5	17	38	80	141	0,1363

Nous remarquons que la distribution des différences doubles égales selon les équations de parité à la deuxième itération n'est ni constante ni linéaire. La figure III.1 nous permet de mieux constater ce phénomène. Ceci fait en sorte que certaines équations de parité seront moins fiables que d'autres et entraîneront une propagation des erreurs plus importantes lors du décodage. Pour enrayer cette propagation

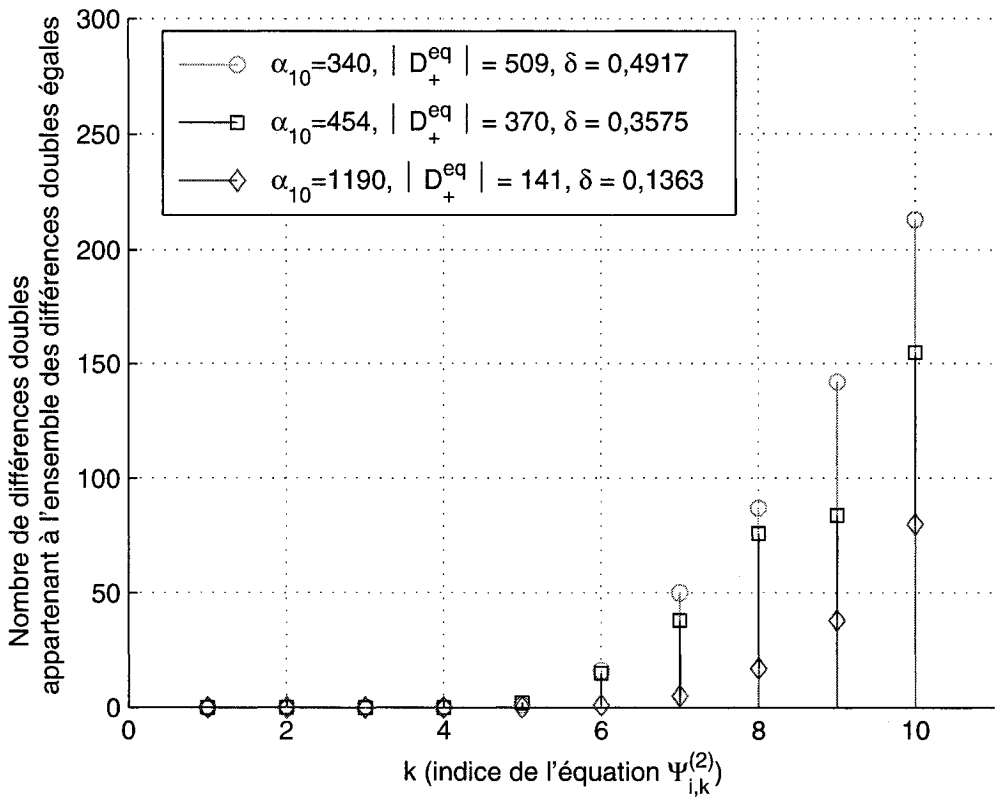

Figure III.1: Nombre de différences doubles par équations de parité appartenant à l'ensemble des différences doubles positives égales , J=10

des erreurs, nous pourrions envisager l'élimination de certaines équations de parité à partir de la seconde itération. Ceci peut se faire en utilisant un coefficient de pondération nul comme nous l'avons décrit à la section 3.4.2. Toutefois, ce compromis n'est pas analysé dans ce mémoire.

Annexe IV

Variation de la longueur des codes simplifiés doublement orthogonaux perforés S-PCSO^{2}C-WS de type EEPP pour différents taux de codage

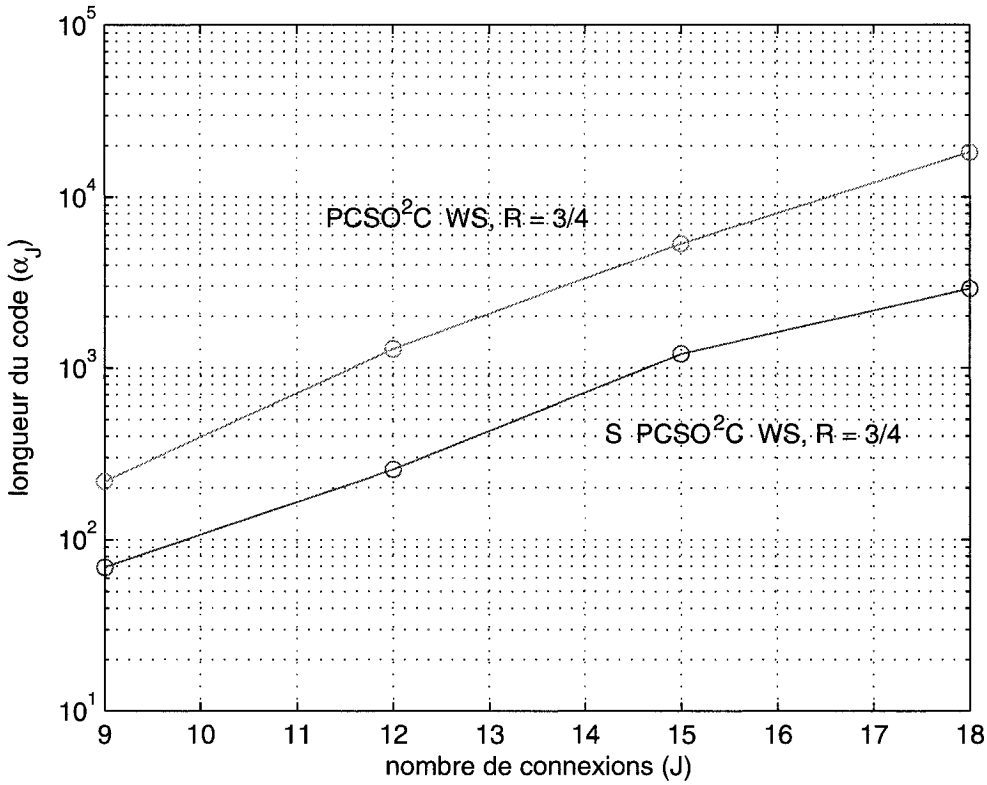

Figure IV.1: Comparaison entre les longueurs les plus courtes pour les codes perforés S-PCSO^{2}C-WS et PCSO^{2}C-WS, $R = \frac{3}{4}$

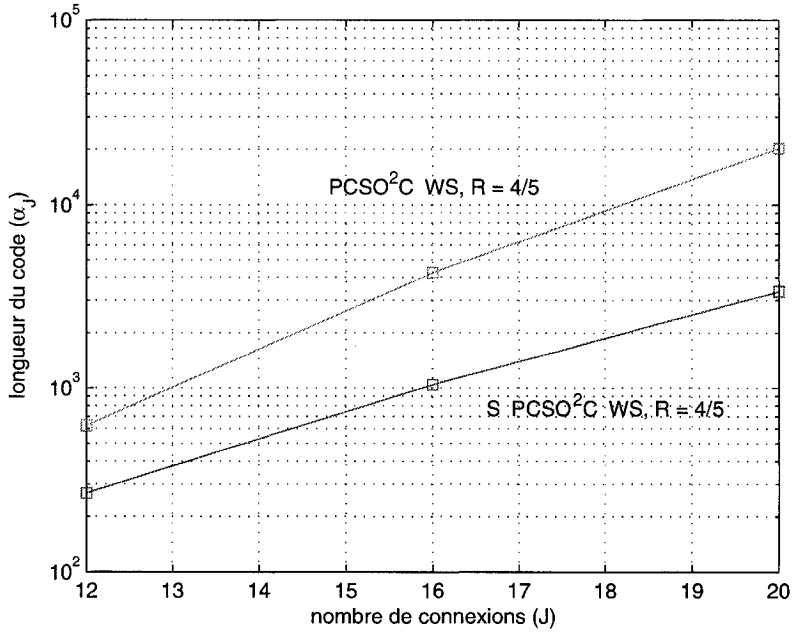

Figure IV.2: Comparaison entre les longueurs les plus courtes pour les codes perforés S-PCSO^{2}C-WS et PCSO^{2}C-WS, $R = \frac{4}{5}$

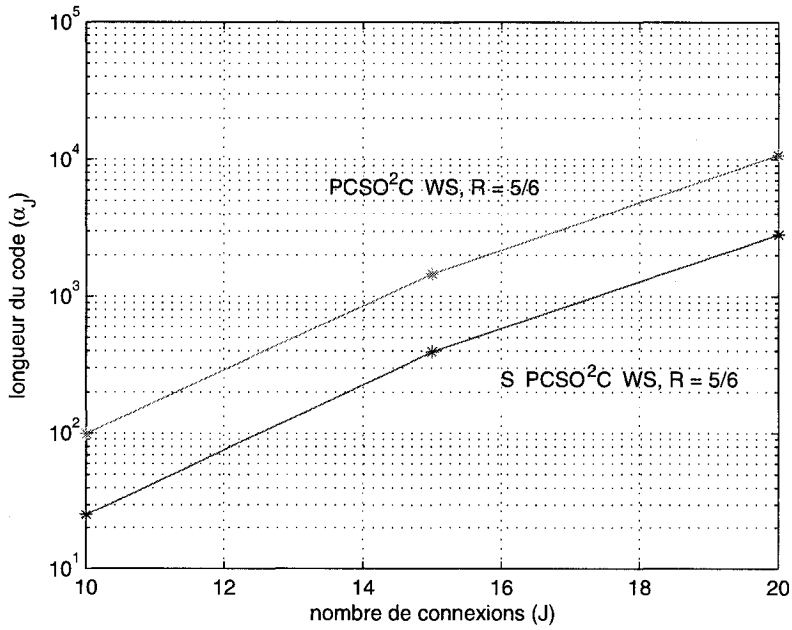

Figure IV.3: Comparaison entre les longueurs les plus courtes pour les codes perforés S-PCSO^{2}C-WS et PCSO^{2}C-WS, $R = \frac{5}{6}$

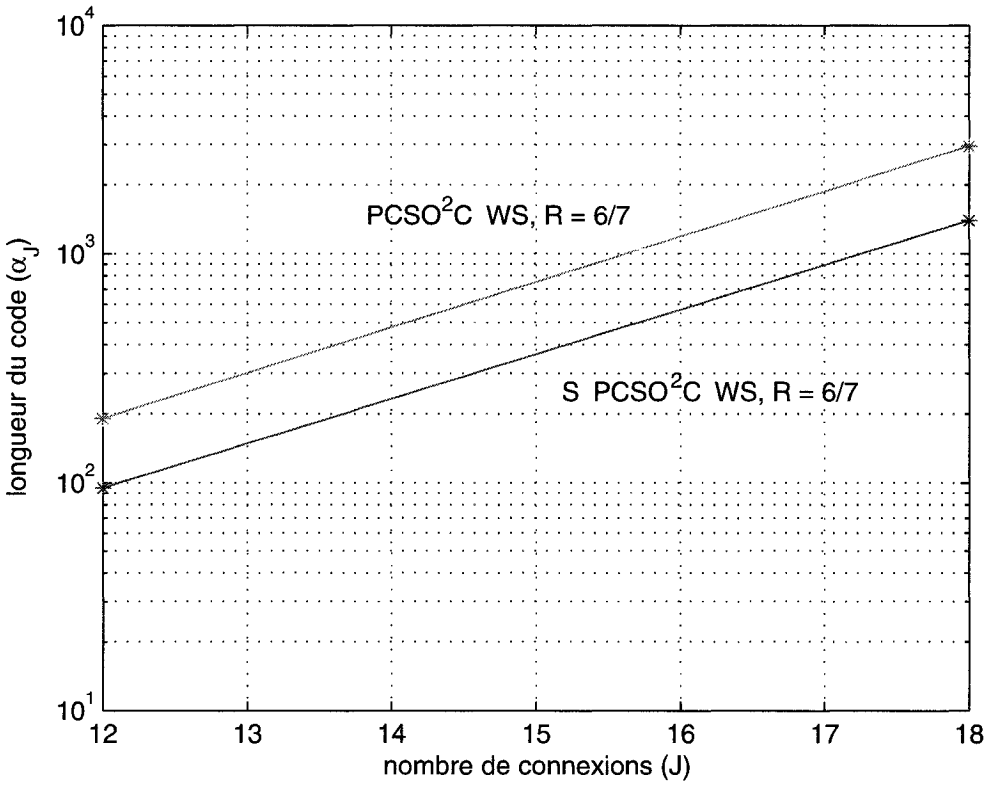

Figure IV.4: Comparaison entre les longueurs les plus courtes pour les codes perforés S-PCSO^{2}C-WS et PCSO^{2}C-WS, $R = \frac{6}{7}$

Annexe V

Performances d'erreur des codes simplifiés au sens large

V.1 Codes S-CSO^{2}C-WS de taux de codage R = $\frac{1}{2}$

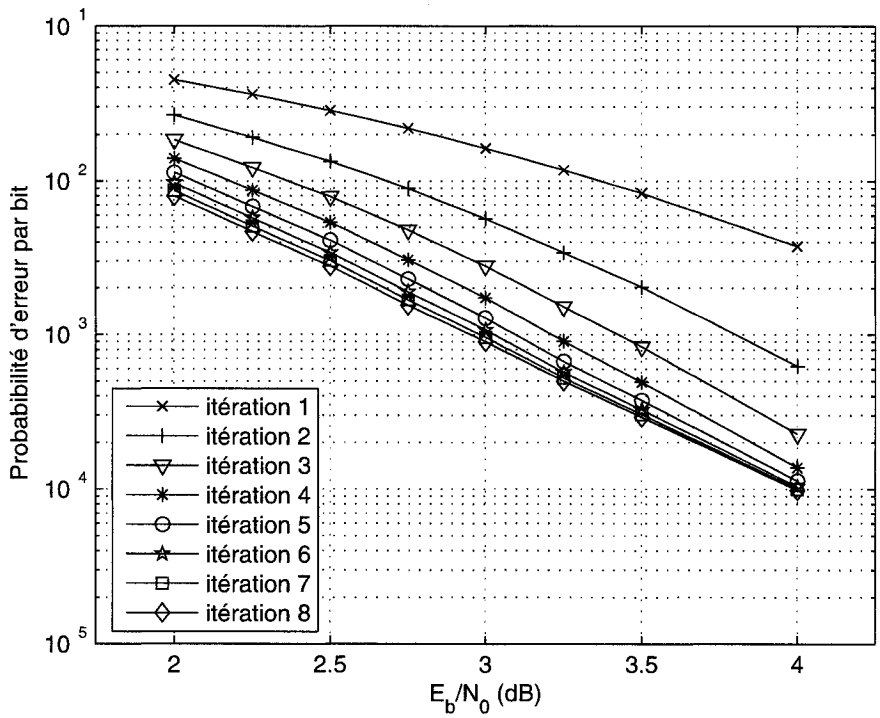

Figure V.1: S-CSO^{2}C-WS, R = 1/2, J=5, $\mathcal{A}$={0, 1, 10, 25, 32}, δ=0.2181

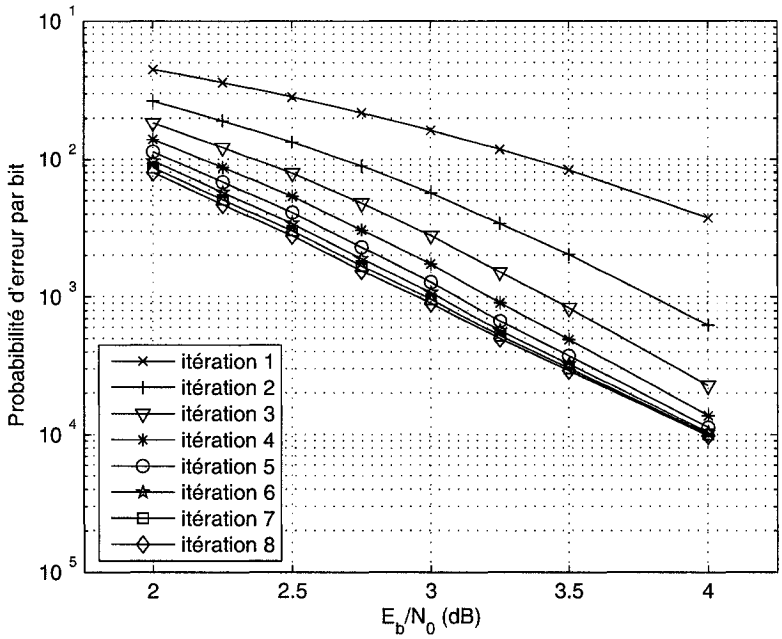

Figure V.2: S-CSO^{2}C-WS, R = 1/2, J=5, $\mathcal{A}$={0, 1, 15, 18, 23}, δ=0.3818

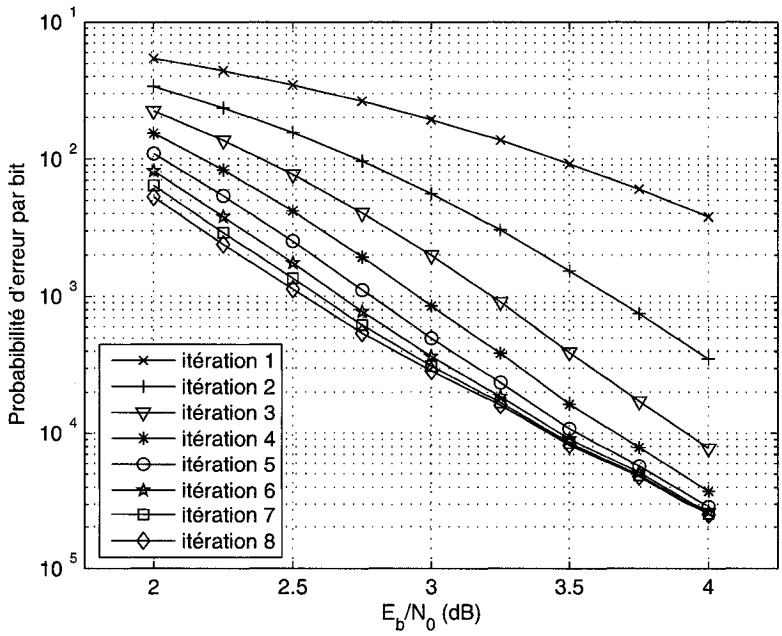

Figure V.3: S-CSO^{2}C-WS, R = 1/2, J=6, $\mathcal{A}$={0, 35, 47, 67, 69, 76}, δ=0.2333

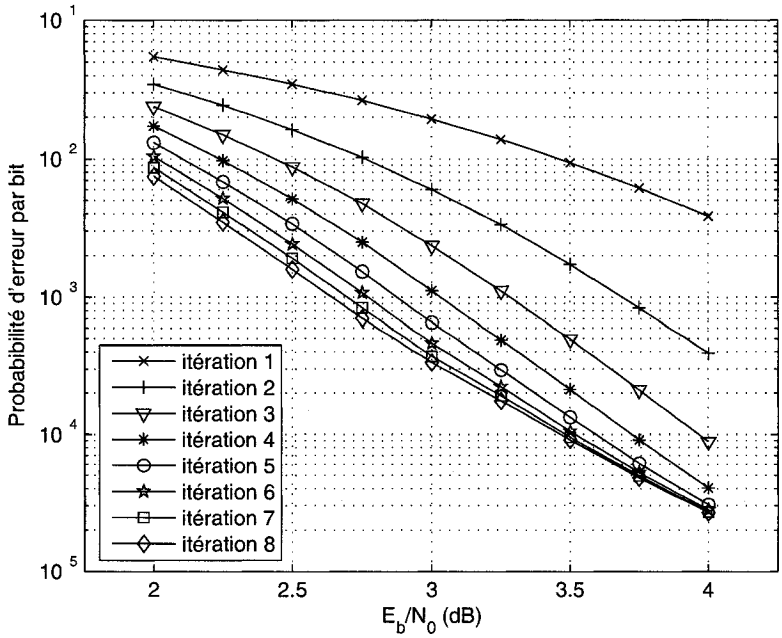

Figure V.4: S-CSO^{2}C-WS, R = 1/2, J=6, $\mathcal{A}$={0, 2, 11, 26, 42, 45}, δ=0.4333

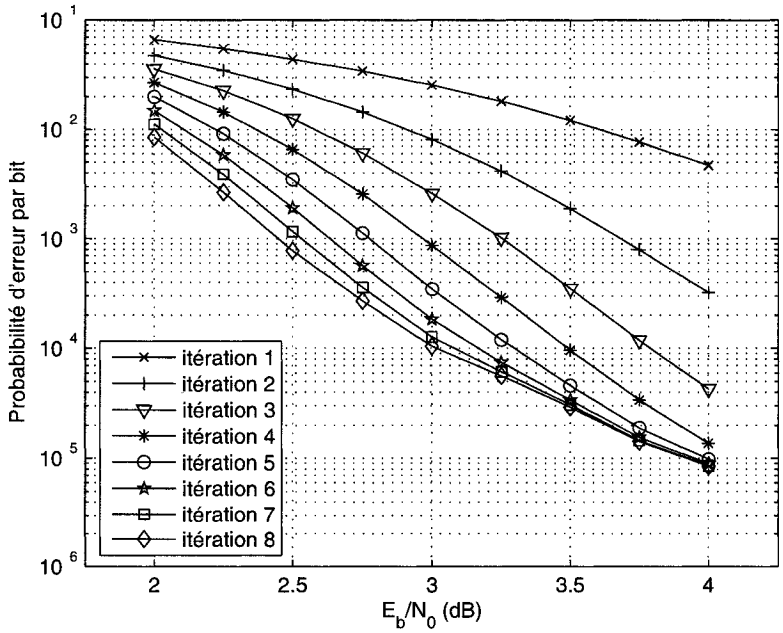

Figure V.5: S-CSO^{2}C-WS, R = 1/2, J=7, $\mathcal{A}$={0, 48, 95, 121, 137, 154, 156}, δ=0.2294

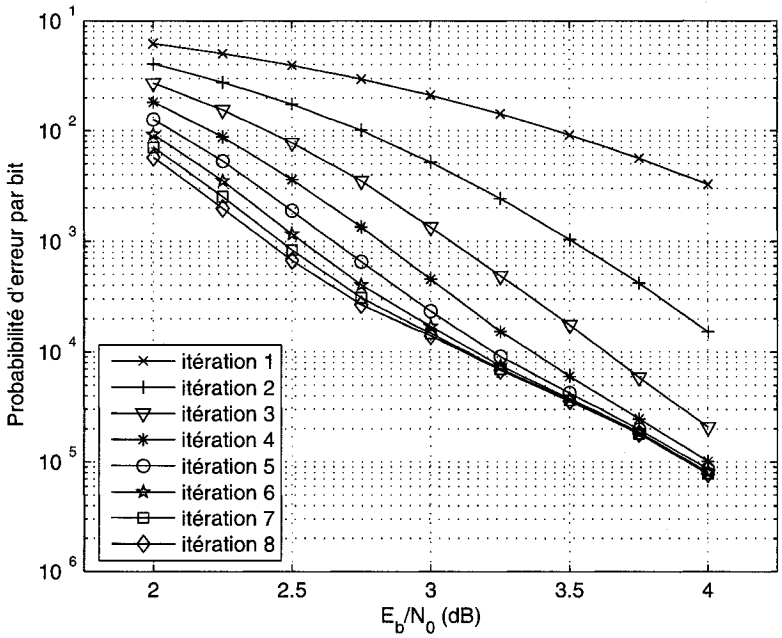

Figure V.6: S-CSO^{2}C-WS, R = 1/2, J=7, $\mathcal{A}$={0, 1, 7, 50, 59, 78, 82}, δ=0.4286

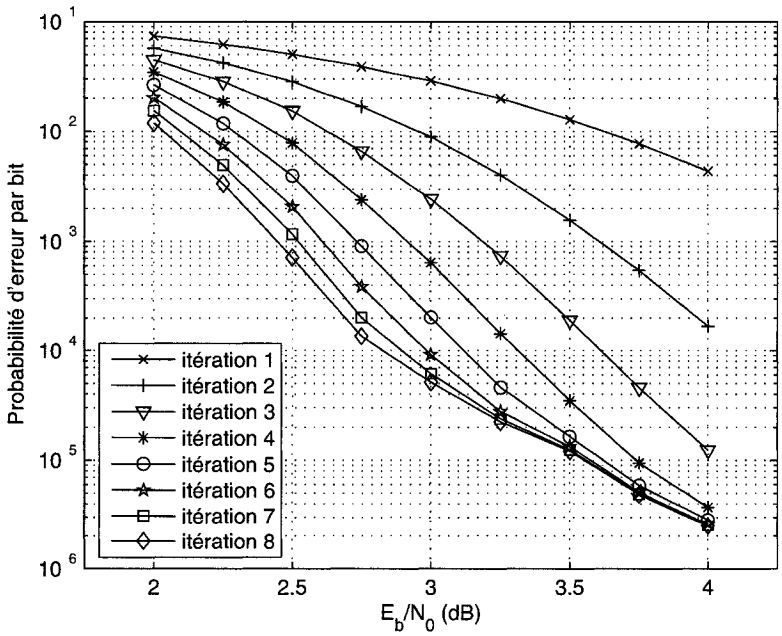

Figure V.7: S-CSO^{2}C-WS, R = 1/2, J=8, $\mathcal{A}$={0, 9, 22, 55, 95, 124, 127, 129}, δ=0.4828

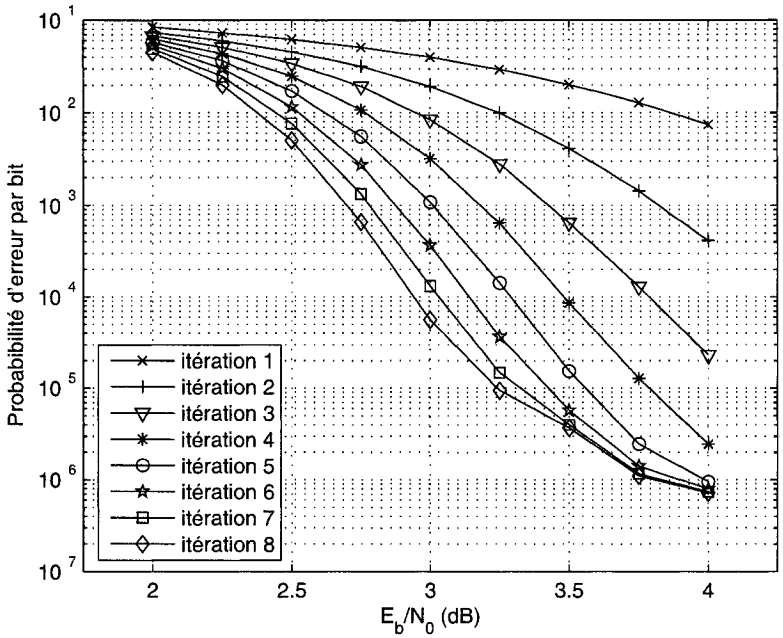

Figure V.8: S-CSO^{2}C-WS, R = 1/2, J=9, $\mathcal{A}$={0, 1, 17, 26, 127, 138, 185, 204, 208}, δ=0.4895

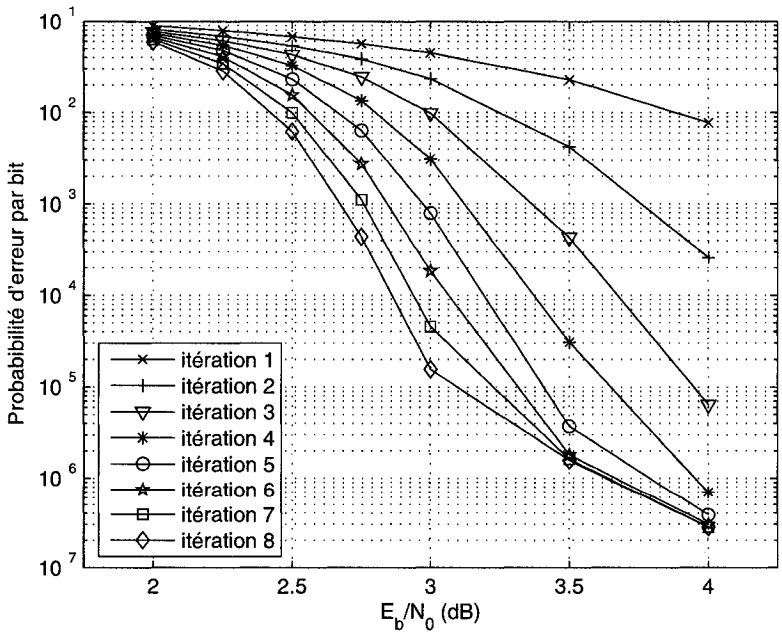

Figure V.9: S-CSO^{2}C-WS, R = 1/2, J=10, $\mathcal{A}$={0, 1, 87, 93, 226, 262, 296, 316, 327, 340}, δ=0.4801

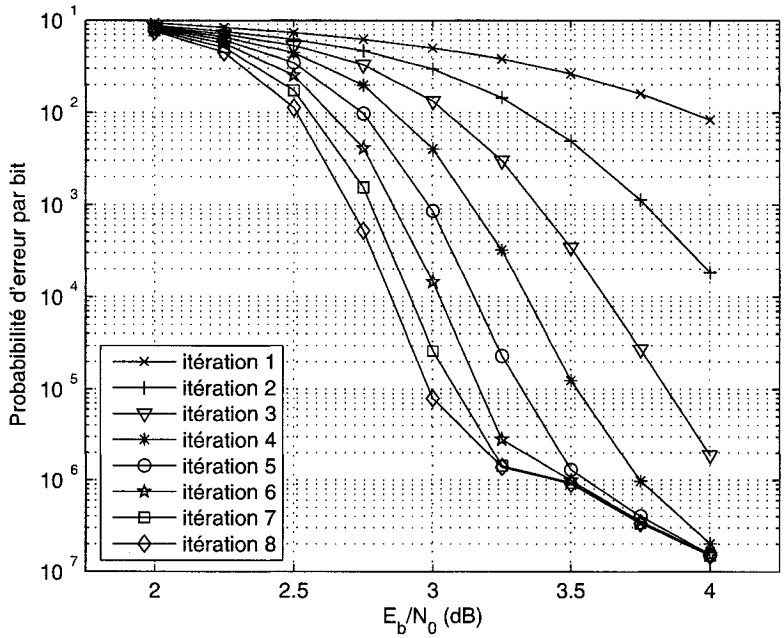

Figure V.10: S-CSO^{2}C-WS, R = 1/2, J=11, $\mathcal{A}$={0, 1, 5, 12, 32, 61, 199, 350, 434, 480, 588 }, δ=0.4539

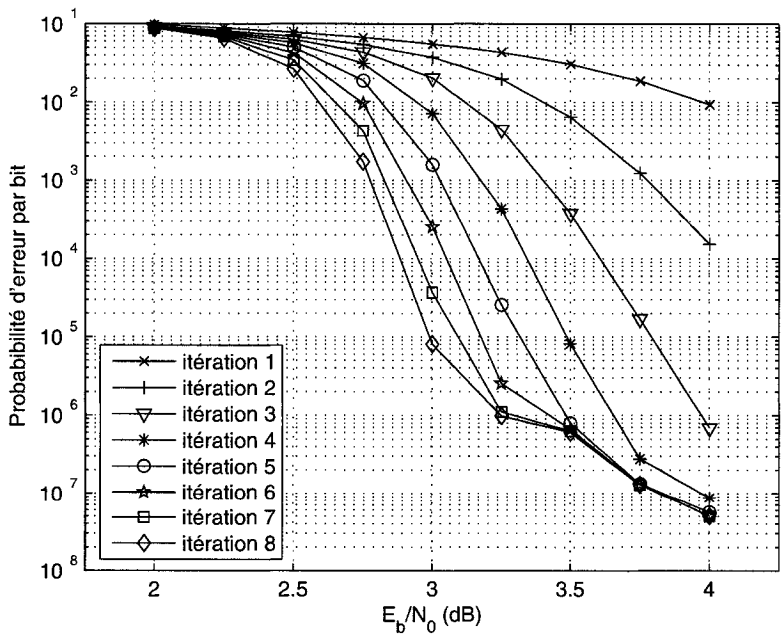

Figure V.11: S-CSO^{2}C-WS, R = 1/2, J=12, $\mathcal{A}$={0, 1, 5, 12, 32, 61, 107, 271, 411, 584, 707, 894}, δ=0.4632

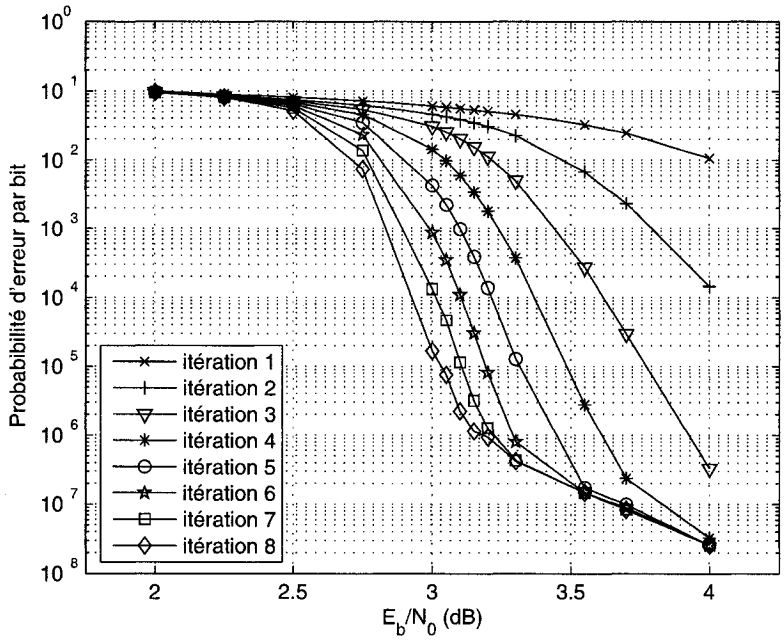

Figure V.12: S-CSO^{2}C-WS, R = 1/2, J=13, $\mathcal{A}$={0, 2, 12, 144, 190, 207, 633, 747, 974, 1052, 1111, 1214, 1217}, δ=0.4193

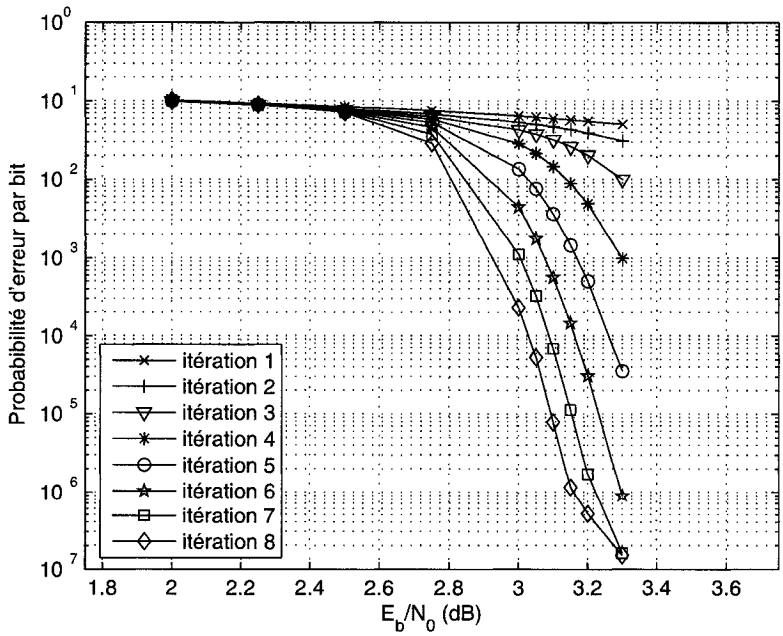

Figure V.13: S-CSO^{2}C-WS, R = 1/2, J=14, $\mathcal{A}$={0, 1, 5, 12, 32, 61, 107, 230, 355, 514, 919, 1188, 1660, 1967}, δ=0.4269

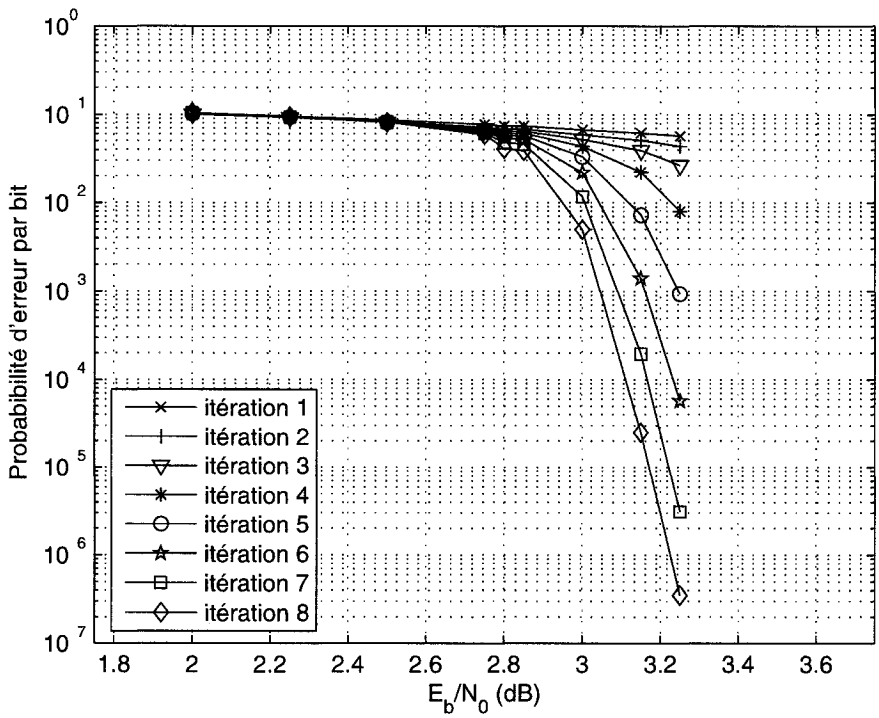

Figure V.14: S-CSO^{2}C-WS, R = 1/2, J=15, $\mathcal{A}$={0, 1, 5, 12, 32, 61, 107, 230, 355, 514, 824, 1424, 1726, 2384, 2653}, δ=0.4253

V.2 Codes S-PCSO^{2}C-WS de taux de codage R = $\frac{2}{3}$

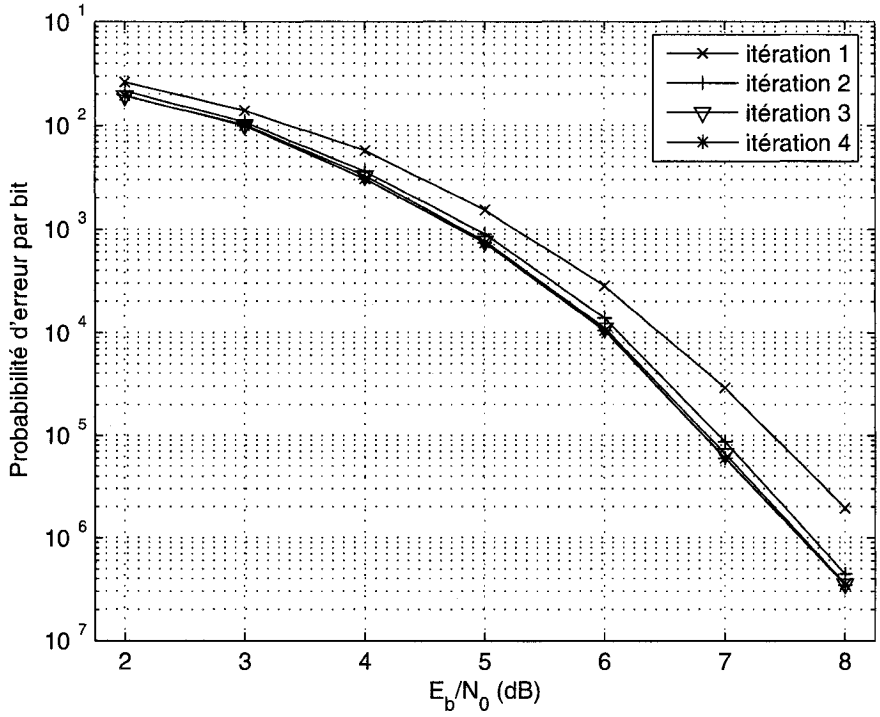

Figure V.15: S-PCSO^{2}C-WS, R = 2/3, J=4, $\mathcal{A}$={0, 3, 10, 11 }

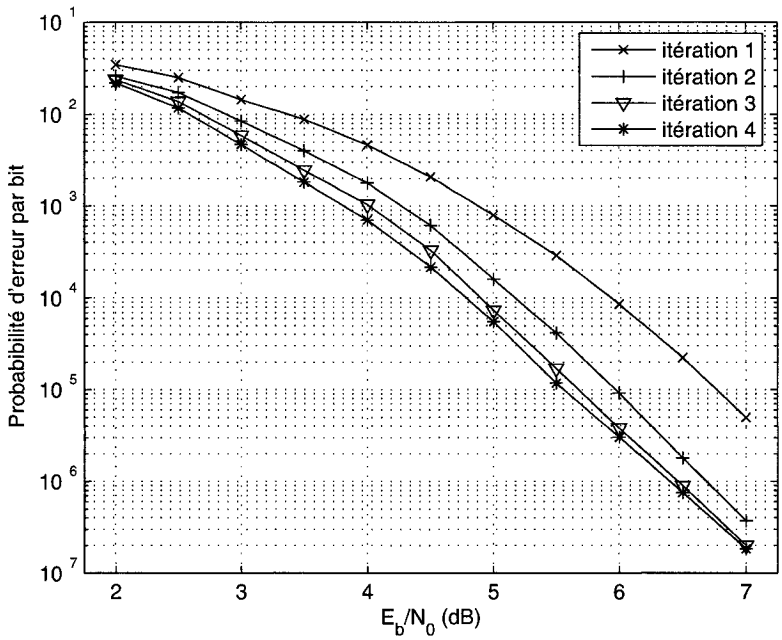

Figure V.16: S-PCSO^{2}C-WS, R = 2/3, J=6, $\mathcal{A}$={0, 3, 10, 11, 17, 22}

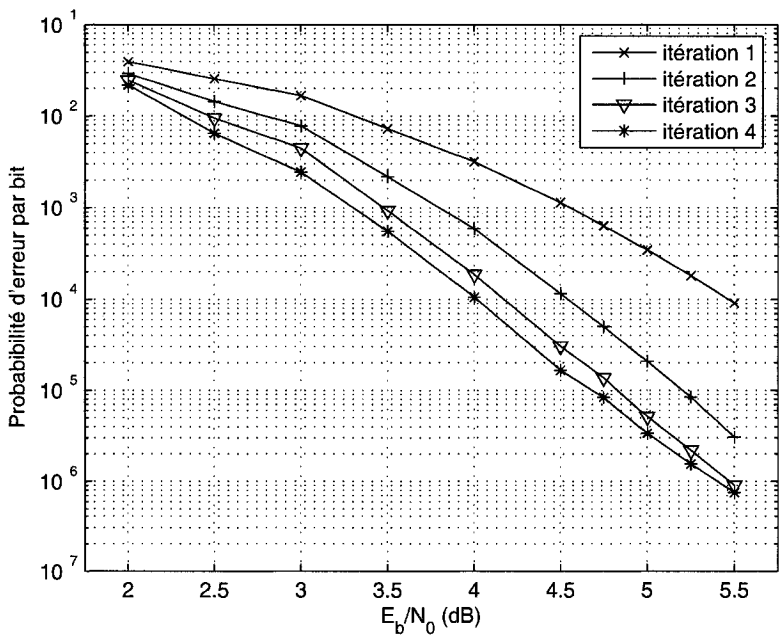

Figure V.17: S-PCSO^{2}C-WS, R = 2/3, J=8, $\mathcal{A}$={0, 5, 12, 31, 51, 66, 67, 68}

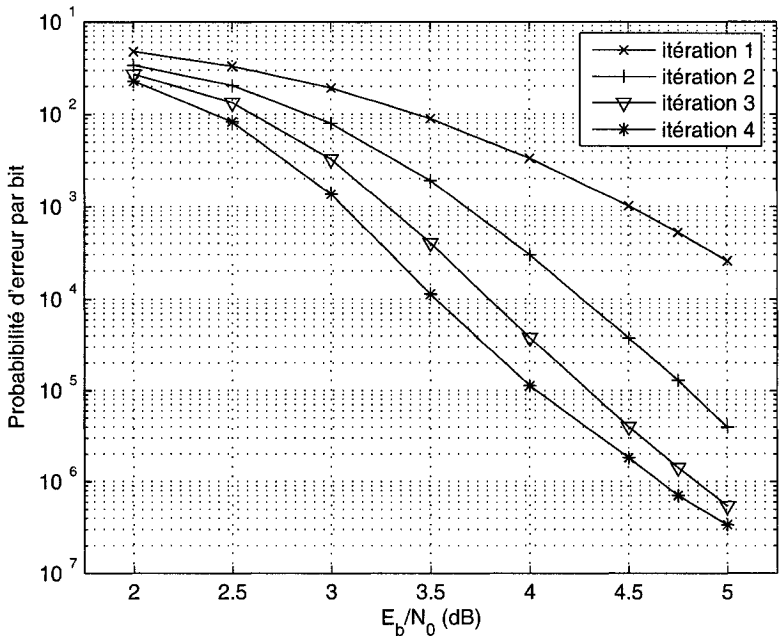

Figure V.18: S-PCSO^{2}C-WS, R = 2/3, J=10, $\mathcal{A}$={0, 30, 51, 97, 117, 160, 214, 221, 225, 232}

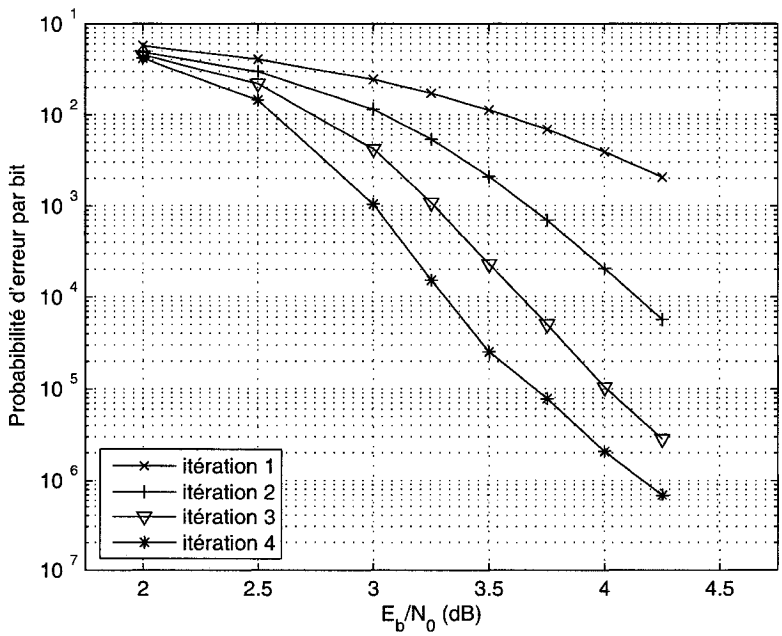

Figure V.19: S-PCSO^{2}C-WS, R = 2/3, J=12, $\mathcal{A}$={0, 18, 93, 106, 293, 497, 827, 922, 948, 954, 967, 977}

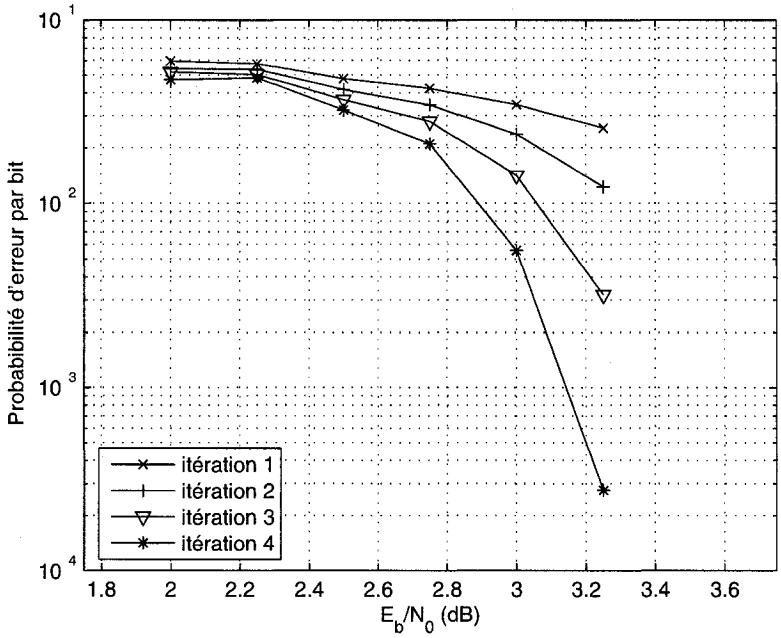

Figure V.20: S-PCSO^{2}C-WS, R = 2/3, J=16, $\mathcal{A}$={0, 97, 477, 490, 958, 1387, 1418, 1455, 1678, 2148, 2411, 2860, 2984, 3015, 3041, 3187}

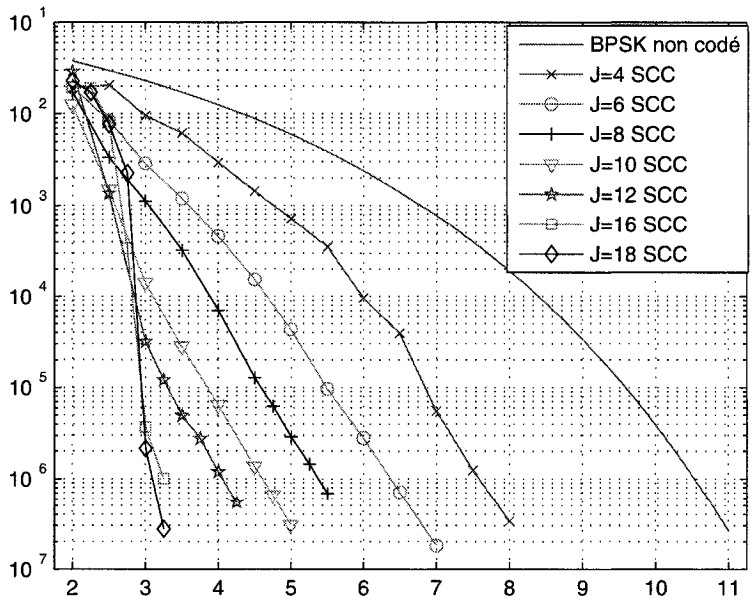

Figure V.21: Probabilité d'erreur par bit pour les codes S-PCSO^{2}C-WS de taux de codage R = 2/3, 8^e itération

V.3 Codes S-PCSO^{2}C-WS de taux de codage R = $\frac{3}{4}$

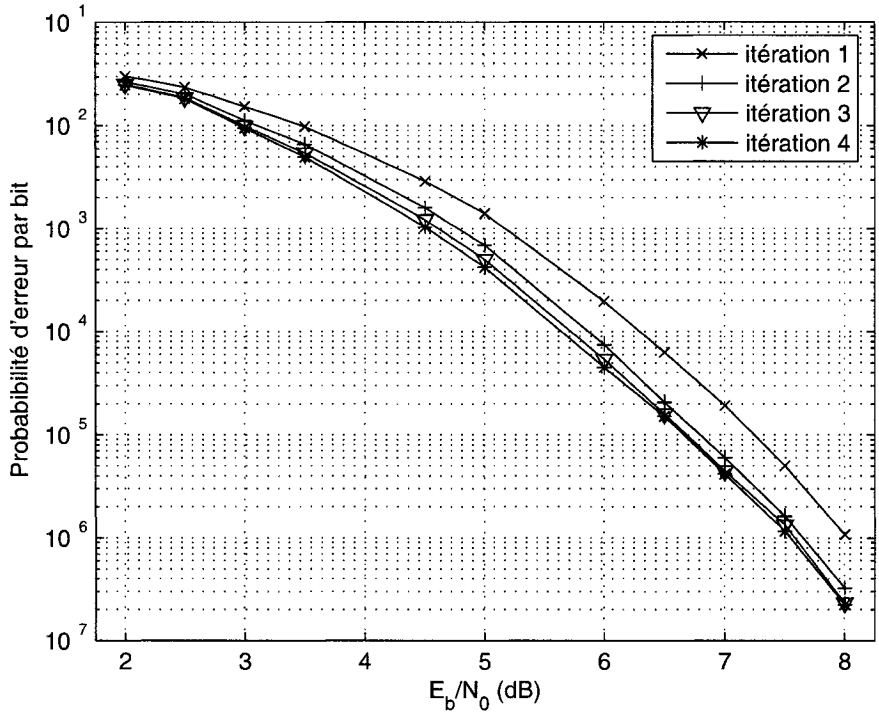

Figure V.22: S-PCSO^{2}C-WS, R = 3/4, J=6, $\mathcal{A}$={ 0, 1, 5, 8, 10, 12}

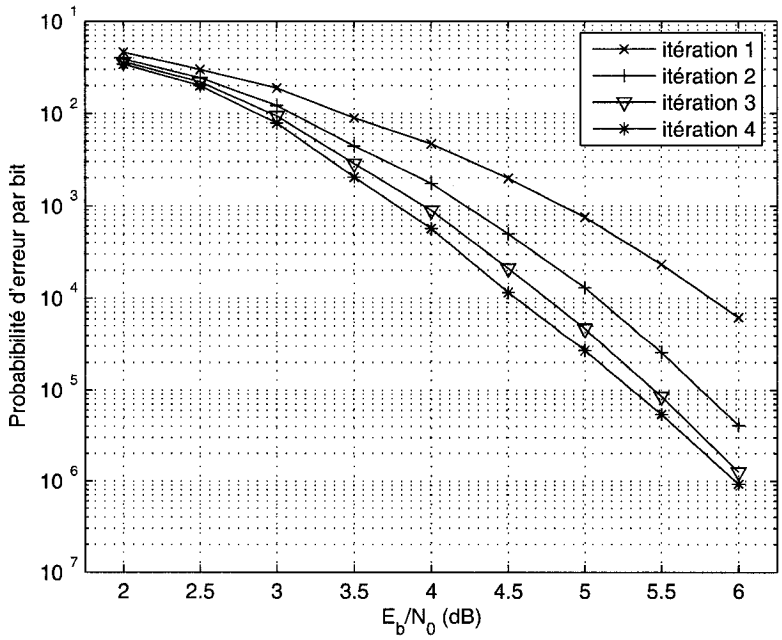

Figure V.23: S-PCSO^{2}C-WS, R = 3/4, J=9, $\mathcal{A}$={ 0, 9, 11, 16, 22, 47, 55, 68, 69}

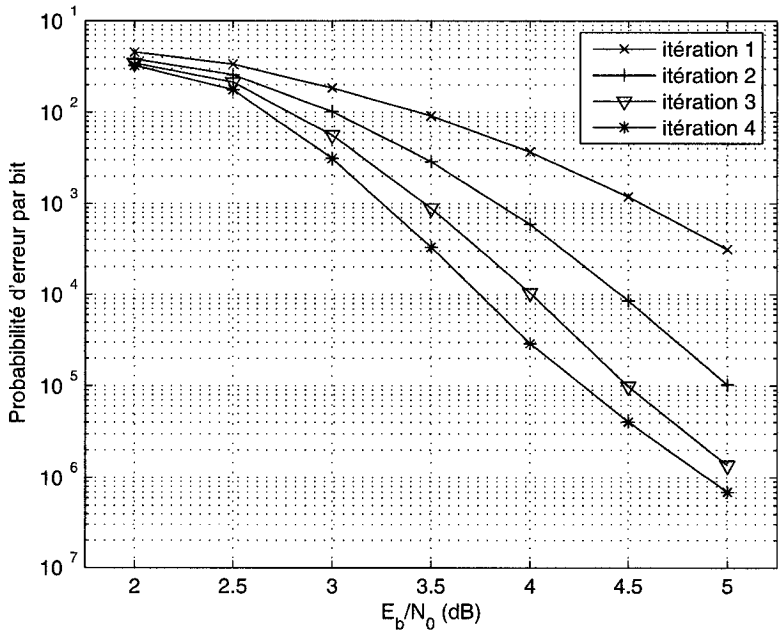

Figure V.24: S-PCSO^{2}C-WS, R = 3/4, J=12, $\mathcal{A}$={0, 32, 83, 156, 176, 178, 273, 310, 340, 343, 347, 348}

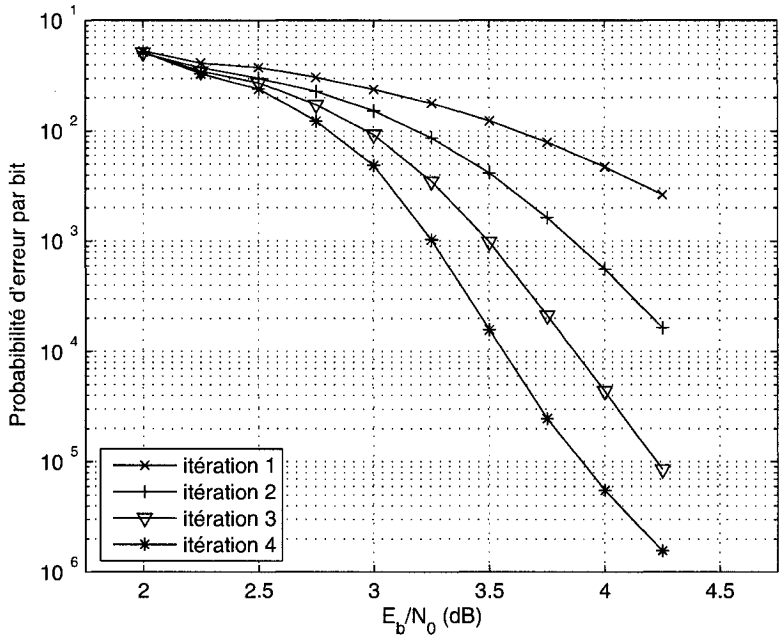

Figure V.25: S-PCSO^{2}C-WS, R = 3/4, J=15, $\mathcal{A}$={ 0, 74, 80, 268, 638, 715, 767, 791, 894, 1386, 1395, 1459, 1501, 1540, 1560}

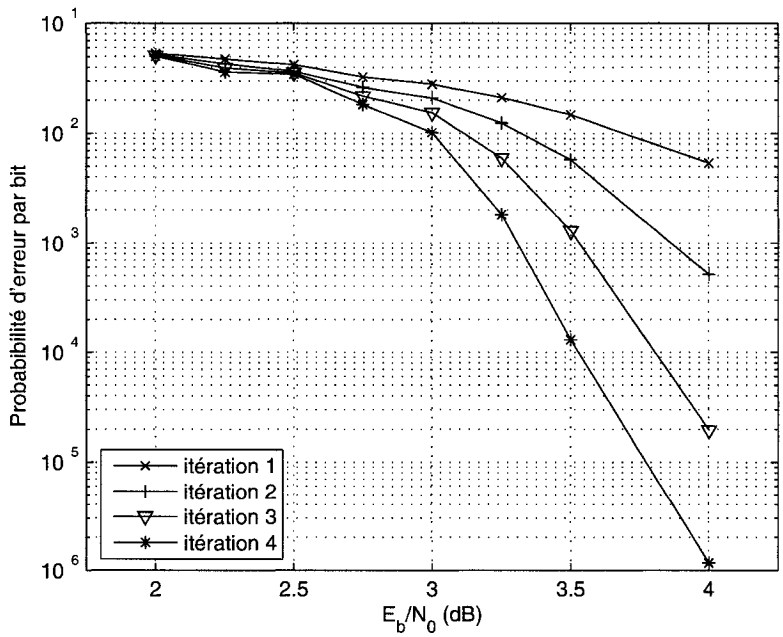

Figure V.26: S-PCSO^{2}C-WS, R = 3/4, J=18, $\mathcal{A}$={0, 62, 101, 483, 817, 1306, 1492, 1737, 1791, 1823, 2004, 2440, 2514, 2681, 2888, 2926, 2930, 2935}

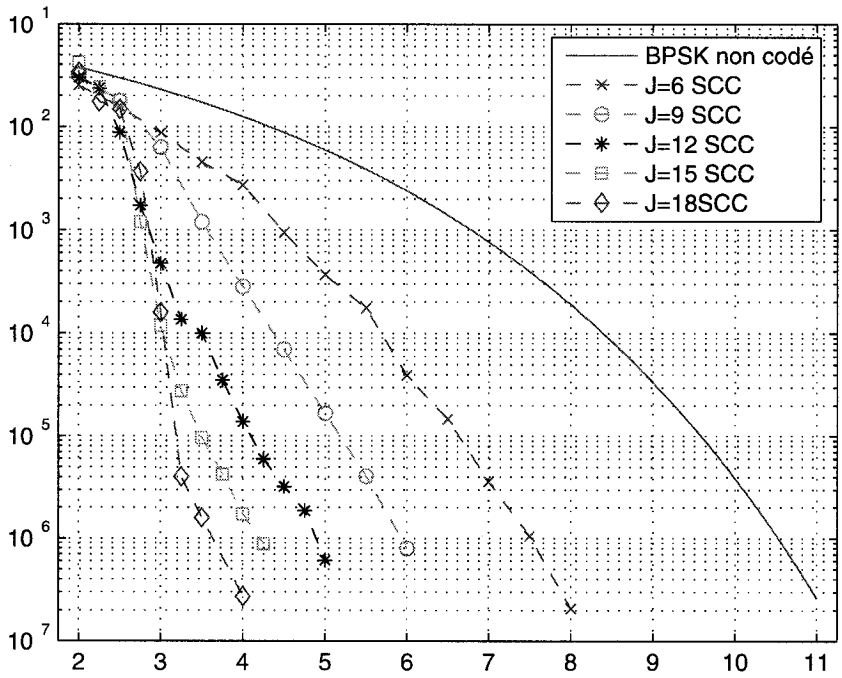

Figure V.27: Probabilité d'erreur par bit pour les codes S-PCSO^{2}C-WS de taux de codage R = 3/4, 8^e itération

V.4 Codes S-PCSO^{2}C-WS de taux de codage R = $\frac{4}{5}$

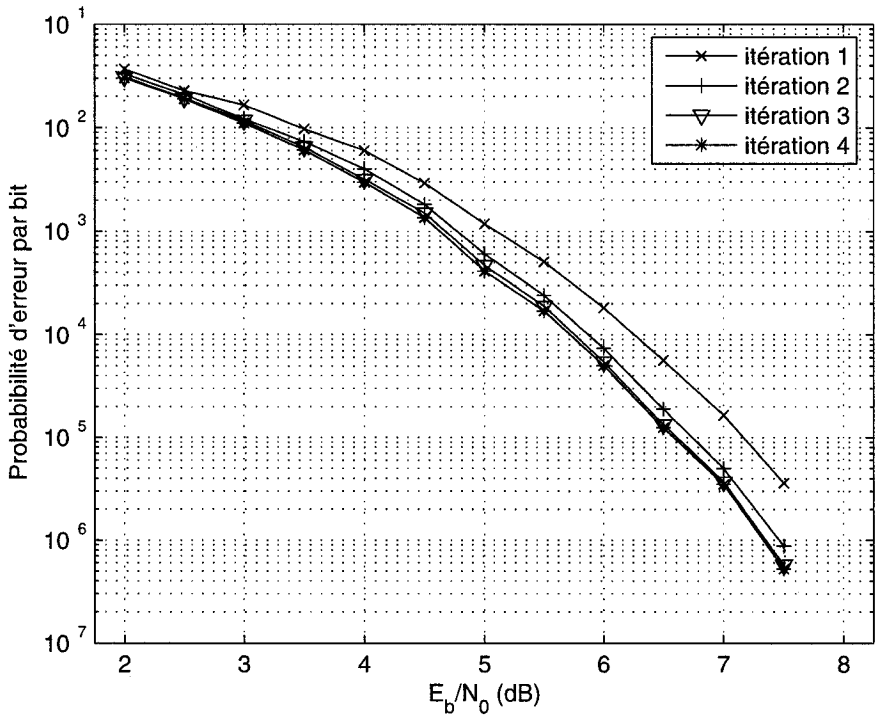

Figure V.28: S-PCSO^{2}C-WS, R = 4/5, J=8, $\mathcal{A}$={0, 1, 2, 3, 7, 9, 14, 20}

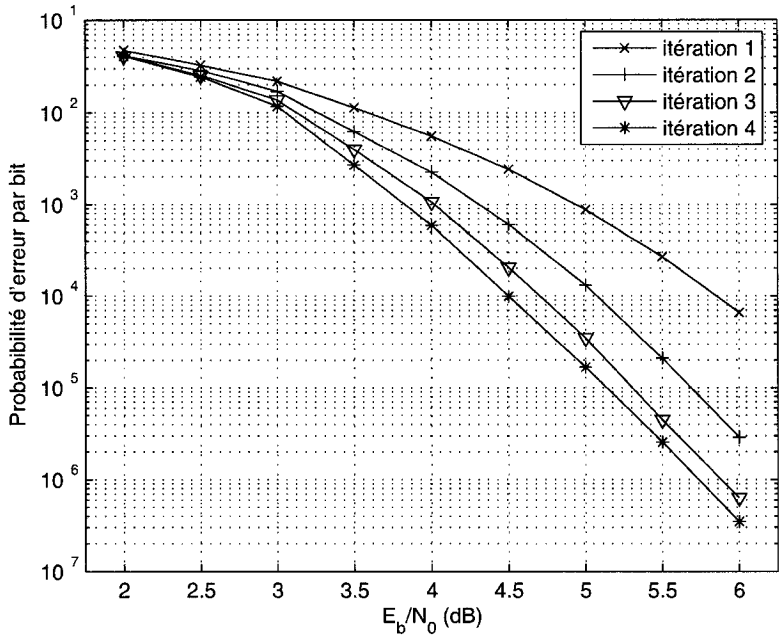

Figure V.29: S-PCSO^{2}C-WS, R = 4/5, J=12, $\mathcal{A}$={0, 19, 92, 165, 168, 173, 226, 301, 307, 314, 342, 363}

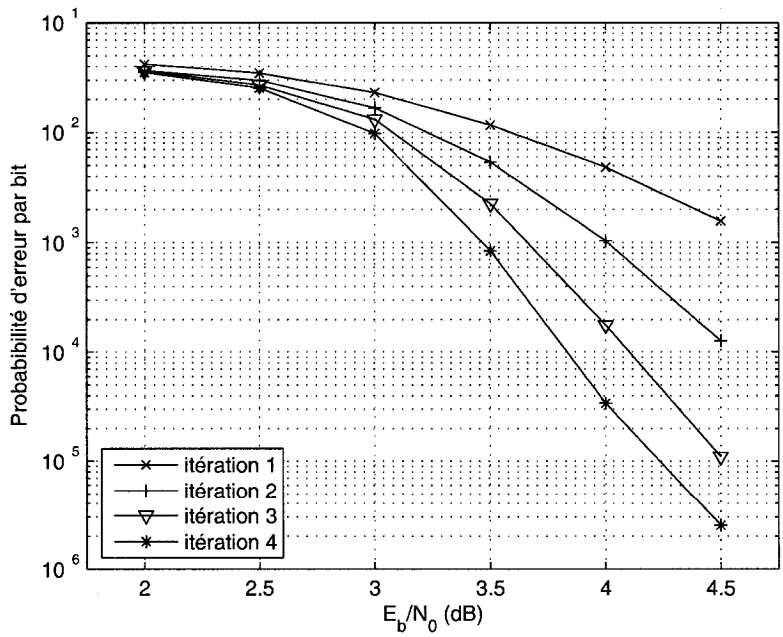

Figure V.30: S-PCSO^{2}C-WS, R = 4/5, J=16, $\mathcal{A}$={0, 171, 334, 365, 367, 640, 715, 734, 895, 1377, 1400, 1510, 1517, 1524, 1530, 1565}

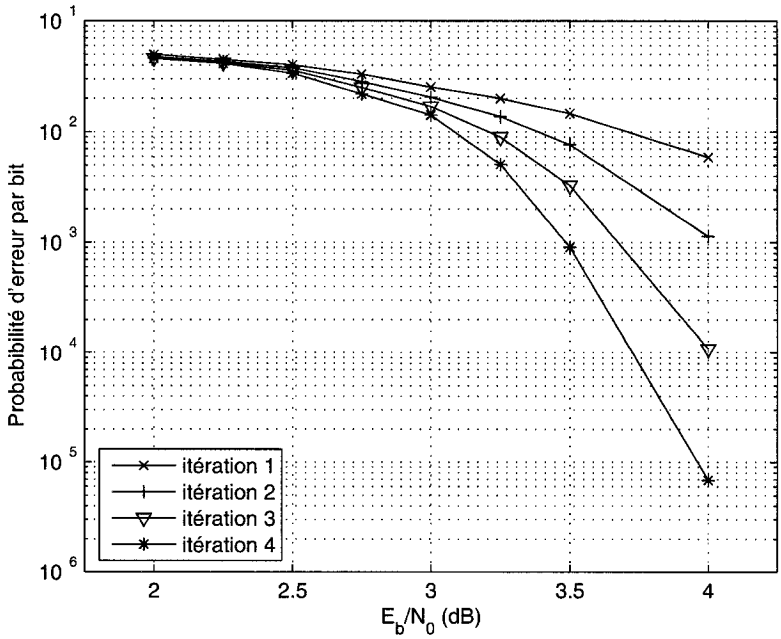

Figure V.31: S-PCSO^{2}C-WS, R = 4/5, J=20, $\mathcal{A}$={0, 60, 509, 986, 994, 1203, 1298, 1702, 1915, 1981, 2084, 2195, 2433, 2463, 2704, 3092, 3265, 3546, 3863, 4017}

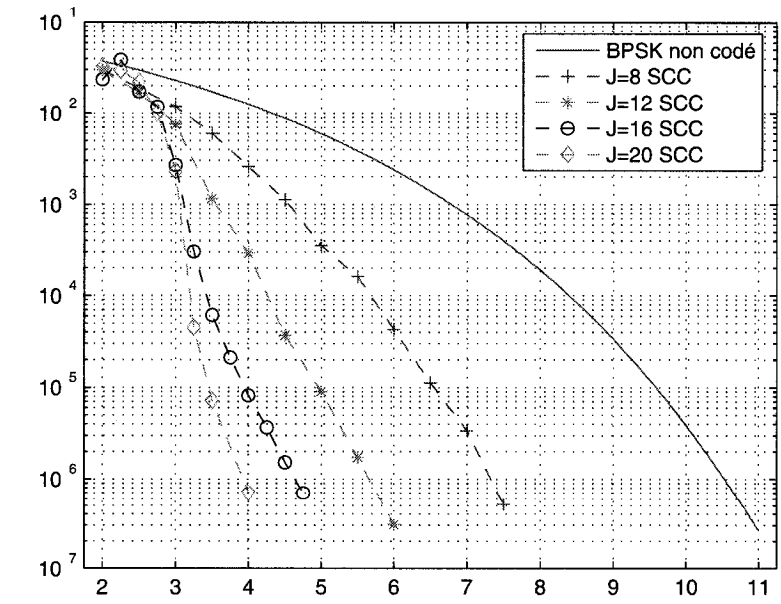

Figure V.32: Probabilité d'erreur par bit pour les codes S-PCSO^{2}C-WS de taux de codage R = 4/5, 8^e itération

V.5 Codes S-PCSO^{2}C-WS de taux de codage R = $\frac{5}{6}$

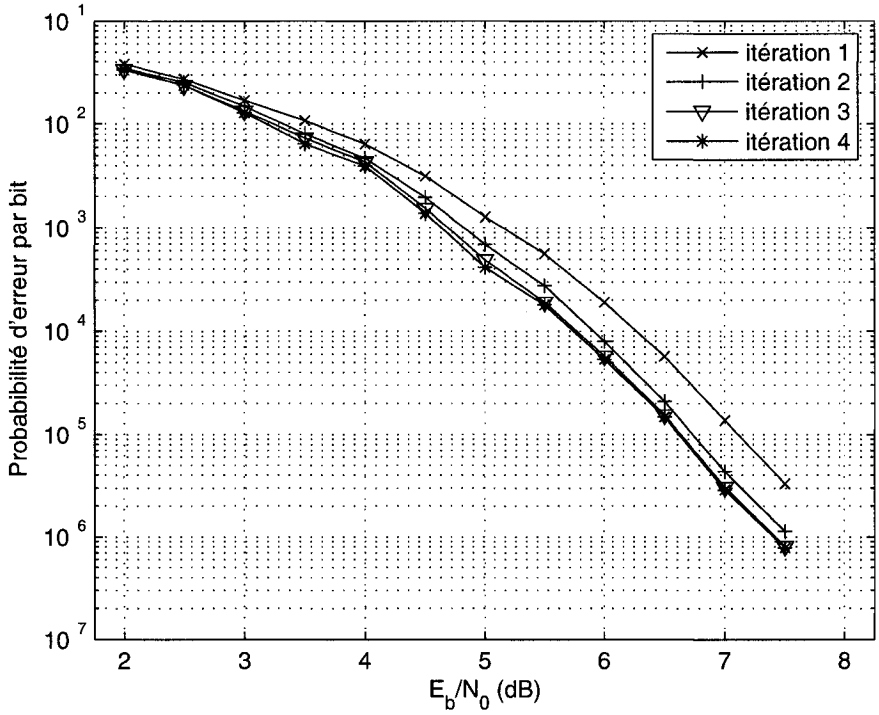

Figure V.33: S-PCSO^{2}C-WS, R = 4/5, J=10, $\mathcal{A}$={0, 3, 6, 7, 9, 14, 17, 21, 23, 25}

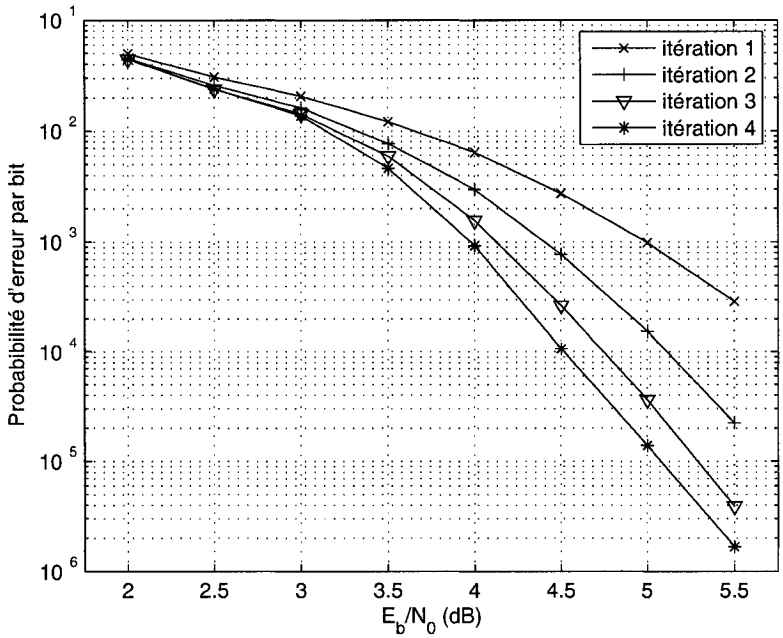

Figure V.34: S-PCSO^{2}C-WS, R = 4/5, J=15, $\mathcal{A}$={0, 63, 205, 206, 359, 400, 408, 457, 874, 883, 901, 917, 919, 921, 922}

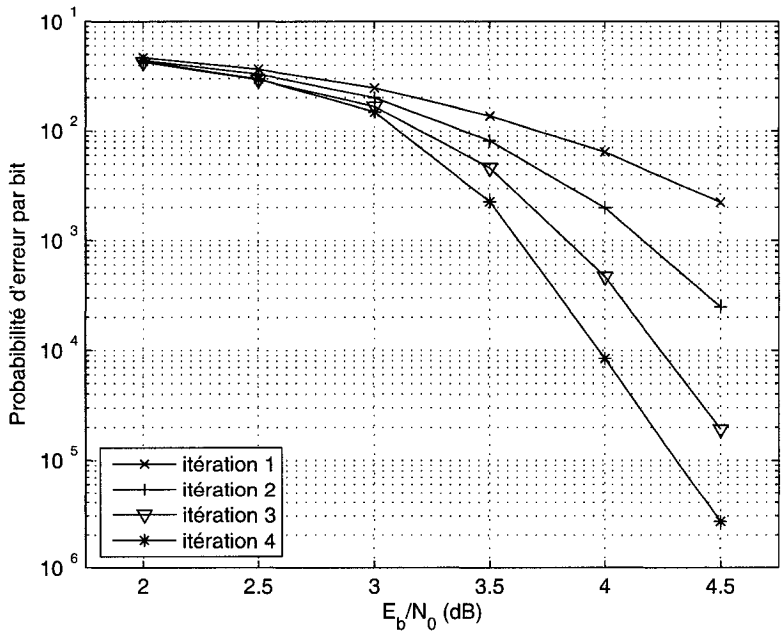

Figure V.35: S-PCSO^{2}C-WS, R = 4/5, J=20, $\mathcal{A}$={0, 33, 68, 140, 632, 921, 941, 1358, 1374, 1418, 1481, 1645, 1857, 2217, 2624, 2676, 2687, 2800, 2829, 2834}

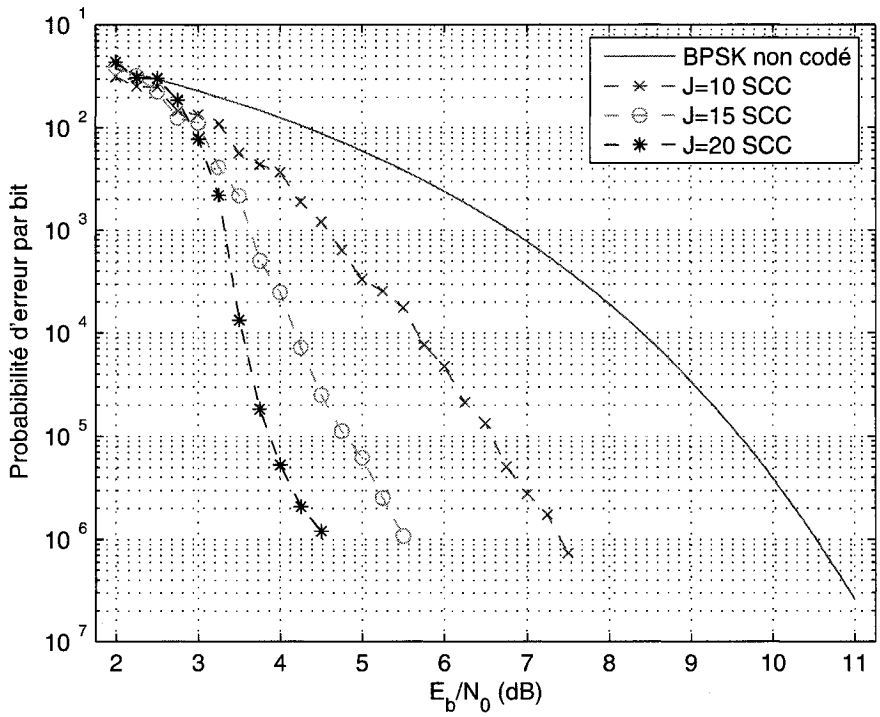

Figure V.36: Probabilité d'erreur par bit pour les codes S-PCSO^{2}C-WS de taux de codage R = 4/5, 8^e itération